2016 IEEE/ACM 13th Working Conference on Mining Software Repositories

Improving Change Recommendation using Aggregated Association Rules

Thomas Rolfsnes* Leon Moonen* Stefano Di Alesio* Razieh Behjati* Dave Binkley‡

thomgrol@simula.no leon.moonen@computer.org stefano@simula.no behjati@simula.no binkley@cs.loyola.edu

* Simula Research Laboratory, Oslo, Norway

‡ Loyola University Maryland, Baltimore, Maryland, USA

ABSTRACT

Past research has proposed association rule mining as a means to uncover the evolutionary coupling from a system's change history. These couplings have various applications, such as improving system decomposition and recommending related changes during development. The *strength* of the coupling can be characterized using a variety of *interestingness measures*. Existing recommendation engines typically use only the rule with the highest interestingness value in situations where more than one rule applies. In contrast, we argue that multiple applicable rules indicate increased evidence, and hypothesize that the aggregation of such rules can be exploited to provide more accurate recommendations.

To investigate this hypothesis we conduct an empirical study on the change histories of two large industrial systems and four large open source systems. As aggregators we adopt three cumulative gain functions from information retrieval. The experiments evaluate the three using 39 different rule interestingness measures. The results show that aggregation provides a significant impact on most measure's value and, furthermore, leads to a significant improvement in the resulting recommendation.

CCS Concepts

•**Software and its engineering** → **Software evolution;** *Software reverse engineering;* •**Information systems** → **Recommender systems;** *Association rules;*

1. INTRODUCTION

As a software system evolves, the number and complexity of interactions in the code grows. For a developer, it becomes increasingly challenging to be in control of the impact of a change made to the system. One potential solution to this problem, *change impact analysis* [7, 13, 21, 30], aims to find artifacts (e.g., files, methods, classes) affected by a given change. This knowledge can then be used either as direct feedback to the developer, or as the basis for another downstream task such as test-case selection and prioritization.

Traditionally, change impact analysis uses static or dynamic dependency analysis [5] (e.g., by identifying the methods that call a changed method). However, in recent years there has been growing interest in alternative approaches. This search is motivated, in part, by limitations in existing techniques. For example, static and dynamic dependency analysis are generally language-specific, making them unsuitable for the analysis of heterogeneous software systems [27]. In addition, they can involve considerable overhead (e.g., dynamic analysis' need for code-instrumentation), and tend to over-approximate the impact of a change [20].

A promising alternative identifies dependencies through *evolutionary coupling*. Such couplings differ from the ones found through static and dynamic dependency analysis, in that they are based on *how* the software was changed over time. In essence, evolutionary coupling taps into the developer's inherent knowledge of the dependencies in the system. This knowledge can manifest itself in several ways, for example, through commit-comments, bug-reports, context-switches in an IDE, etc. In this paper we consider *co-change* as the basis for uncovering evolutionary coupling. Co-change information can, for example, be extracted from a project's version control system [8], from its issue tracking system, or by instrumenting the development environment [22].

The dominant method for mining evolutionary coupling from co-change data is *association rule mining* [1]. The resulting mined rules are often characterized using a variety of *interestingness measures* that aim to capture how informative each rule is relative to the others [14, 16]. Existing approaches that use mined rules to provide change recommendations generally consider only the rule with the highest interestingness value in situations where multiple rules can be applied [26]. In contrast, we conjecture that having multiple applicable rules should be seen as corroborative evidence that the rules capture an important relation or conclusion, and hypothesize that the aggregation of such rules can be exploited to provide more accurate recommendations.

Contributions: This paper presents three key contributions: (1) We investigate a previously unexplored area in association rule mining by considering rule aggregation. (2) We present and formalize *hyper-rules* as a novel approach to aggregate multiple association rules. (3) We evaluate the viability of hyper-rules in the context of change recommendation using a large empirical study of four open source systems as well as two systems from our industry partners.

MSR'16, May 14-15, 2016, Austin, TX, USA

DOI: http://dx.doi.org/10.1145/2901739.2901756

Overview: Our presentation of aggregated association rules is organized as follows: Section 2 provides background on targeted association rule mining. Section 3 describes limitations of classical approaches. Section 4 overviews the forty interestingness measures used to weight the mined association rules. Section 5 introduces the notion of a hyper-rule. Section 6 presents the three aggregation functions considered in the experiments. Section 7 describes the setup of our empirical investigation whose results are presented in Section 8. Finally, Section 9 discusses the implications of our results, Section 10 presents the related work, and then Section 11 provides some concluding remarks.

2. ASSOCIATION RULE MINING

Agrawal et al. introduced the concept of *association rule mining* as the discipline aimed at inferring relations between *entities* of a dataset [1]. *Association rules* are implications of the form $A \rightarrow B$, where A is referred to as the *antecedent*, B as the *consequent*, and A and B are disjoint sets. For example, consider the classic application of analyzing shopping cart data: if multiple transactions include bread and butter then a potential association rule is *bread* $\rightarrow$ *butter*. This rule can be read as *"if you buy bread, then you are also likely to buy butter."*

In the context of mining evolutionary coupling from co-change information, the entities are the files of the system[1] and the collection (history) $\mathcal{T}$ of transactions, is the set of past *commits*. More specifically, a transaction $T \in \mathcal{T}$ is the set of files that were either changed or added while addressing a given bug or feature addition, hence creating a *logical dependence* between the files [9].

As originally defined [1], association rule mining generates rules that express patterns in a complete data set. However, some applications can exploit a more focused set of rules. *Targeted association rule mining* [24] focuses the generation of rules by applying a constraint. One example constraint specifies that the antecedent of all mined rules belongs to a particular set of files, which effectively reduces the number of rules that need to be created. This reduction drastically improves the execution time of rules generation [24].

When performing change impact analysis, rule-constraints are based on a *change set*, e.g., the set of modified files since the last commit. In this case, only rules with at least one changed entity in the antecedent are created. The output of change impact analysis are the files from the system that are historically changed alongside the elements of the change set. For example, given the change-set $\{a, b, c\}$, change impact analysis would uncover files that were changed when a, b, and c were changed. The resulting impacted files are those found in the rule consequents. These files can be ranked based on the rule's interestingness-measure.

To our knowledge, only a few targeted association rule mining algorithms have been considered in the context of change impact analysis: Zimmerman et al. [31], Ying et al. [28], and Rolfsnes et al. [23] (our previous work). In contrast, simple *co-change* algorithms have been well studied in a variety of contexts [2, 4, 9, 11]. The existing targeted association rule mining algorithms and the simple co-change algorithms differ in terms of which subsets of the change-set that are allowed in the antecedent of generated rules. Consider, for example, the subsets of the change-set $C = \{a, b, c, d\}$:

$$
\begin{aligned}
powerset(C) = \{\{\}, & && (1)\\
& \{a\}, \{b\}, \{c\}, \{d\}, && (2)\\
& \{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}, \{c, d\}, && (3)\\
& \{a, b, c\}, \{a, b, d\}, \{a, c, d\}, \{b, c, d\}, && (4)\\
& \{a, b, c, d\}\} && (5)
\end{aligned}
$$

Of C's subsets, both Zimmerman's and Ying's algorithms only consider rules based on line 5 (i.e., rules of the form $\{a, b, c, d\} \rightarrow X$) because these techniques constrain the antecedent to be equal to the change set. At the other end of the spectrum, co-change algorithms consider rules from the singleton sets in line 2, such as $a \rightarrow X$ or $b \rightarrow X$. Rolfsnes et al. introduce TARMAQ, the most versatile among these existing algorithms. TARMAQ can use the sets from any of lines 2, 3, 4, or 5. The particular line used is dynamically chosen based on the maximal overlap with the change set [23].

3. PROBLEM DESCRIPTION

Change impact analysis takes as input a set of (recently) changed entities, referred to as a *change set*, and outputs a set of potentially impacted entities. The application of association rule mining to change impact analysis involves looking for the *evolutionary coupling* between entities (hereafter files) of a system. This search considers files coupled iff they have changed together in the past. In addition, it is valuable to capture the *strength* of a coupling, which is stronger the more frequently the files change together.

In our previous work [23], which sought to find evolutionary couplings through association rule mining of a system's version history, we noticed that there are often rules with different *antecedents*, but the same *consequent*. For example, consider the following rules involving files a, b, and c:

$$
\begin{aligned}
r_1 &= \{a\} \rightarrow \{c\}\\
r_2 &= \{b\} \rightarrow \{c\}
\end{aligned}
$$

which can be interpreted as "if you change a, consider changing c," and "if you change b consider changing c." Given the change set $\{a, b\}$, existing recommendations systems will choose one of the two rules, that recommend c. However, we argue that using only one of the two rules can be a mistake, since having multiple applicable rules for the same consequent potentially provides increased evidence that the consequent is relevant. We hypothesize that this increased evidence can be captured by the *aggregation* of rules into *hyper-rules* that produce more accurate recommendations. In other words, we seek to combine rules r_1 and r_2 into the *hyper-rule* r_3 that captures the cumulative evidence that c should be recommended for change when a and b are changed.

A concrete example will help illustrate our goal and also provide a better intuition into the value of hyper-rule formation. The example involves a sequence of past transactions that each include a set of files that changed together. The example also motivates the need to aggregate the *interestingness values* of the rules producing an interestingness value for the resulting hyper-rule. The example does this

[1] Observe that other levels of granularity are possible, and our consideration of co-change at the file level is without loss of generality, as these algorithms are granularity agnostic: provided that suitably co-change data is available (or computable), the algorithms will relate methods or variables just as well as files.

using, as a simple interestingness value, the percentage of the transactions that give rise to the rule.

Example 1 *Consider the following (historic) sequence of transactions:*

$$\mathcal{T} = [\{a, x\}, \{b, y\}, \{c, y\}, \{d, y\}, \{a, x\}]$$

and the change set $C = \{a, b, c, d\}$ where, based on $\mathcal{T}$ and C, the following rules have been mined (the interestingness of each is given in parentheses):

$$a \rightarrow x \quad (40\%)$$
$$b \rightarrow y \quad (20\%)$$
$$c \rightarrow y \quad (20\%)$$
$$d \rightarrow y \quad (20\%)$$

In these rules all the files that occur in an antecedent are part of change set C while all files that occur in a consequent are potentially impacted by the change with a certainty reflected by the rule's interestingness value.

In Example 1, without aggregation, x is recommended above y, because it has changed two times with an item in the change set (a), while y had changed at most once with *any individual* item of the change set. However, y has changed more times with *at least one* item of the change set than x. Therefore, there is combined evidence that y should be recommended above x.

Generalizing this example, our goal is to aggregate mined association rules into hyper-rules that combine evidence and ultimately provide more accurate recommendations. To this end, the remainder of this paper investigates the impact of three aggregation techniques on the performance of two association rule mining algorithms using a collection of forty interestingness-measures.

4. INTERESTINGNESS MEASURES

The relative value of the rules mined by a targeted association rule mining algorithm is given by an *interestingness measure*. In Agrawal et al.'s seminal paper on association rule mining [1], two predominant interestingness measures are introduced, *support* and *confidence*. Given a set of transactions $\mathcal{T}$, the *support* of the rule $A \rightarrow B$ is defined as the number of transactions where the union of the antecedent and consequent is a subset, divided by the total number of transactions:

$$support(A \rightarrow B) \stackrel{\text{def}}{=} \frac{|\{T \in \mathcal{T} : \{A \cup B\} \subseteq T\}|}{|\mathcal{T}|} \qquad (6)$$

Intuitively, higher support for a rule means that it is more likely to hold. Alternatively, rules with low support identify weaker relations. For this reason, a minimum threshold on support is often used to filter out uninteresting rules.

Second, *confidence* is defined as the number of transactions where the union of A and B is a subset, divided by the number of transactions where A is a subset. It thus gives the conditional probability of B being a subset, given that A is a subset. Formally,

$$confidence(A \rightarrow B) \stackrel{\text{def}}{=} \frac{|\{T \in \mathcal{T} : \{A \cup B\} \subseteq T\}|}{|\{T \in \mathcal{T} : A \subseteq T\}|} \qquad (7)$$

Since the introduction of targeted association rule mining, numerous alternative interestingness measures have been proposed, too many to detail here. Common to all however, is the same set of basic probabilistic measures. Given the rule $A \rightarrow B$, an interestingness measure can be defined in terms of the probabilities of A, B, $\neg A$, $\neg B$ and various combinations obtained through unions, fractions, etc.

Table 1: Overview of probabilistic building blocks where for the rule $A \rightarrow B$ X can be A or B and Y the other

Probability	Definition
$P(X)$	$\frac{\lvert\{T \in \mathcal{T} : X \subseteq T\}\rvert}{\lvert\mathcal{T}\rvert}$
$P(\neg X)$	$1 - P(X)$
$P(X, Y)$	$\frac{\lvert\{T \in \mathcal{T} : \{X \cup Y\} \subseteq T\}\rvert}{\lvert\mathcal{T}\rvert}$
$P(\neg X, \neg Y)$	$\frac{\lvert\{T \in \mathcal{T} : X \not\subseteq T \wedge Y \not\subseteq T\}\rvert}{\lvert\mathcal{T}\rvert}$
$P(\neg X, Y)$	$\frac{\lvert\{T \in \mathcal{T} : X \not\subseteq T \wedge Y \subseteq T\}\rvert}{\lvert\mathcal{T}\rvert}$
$P(X, \neg Y)$	$\frac{\lvert\{T \in \mathcal{T} : X \subseteq T \wedge Y \not\subseteq T\}\rvert}{\lvert\mathcal{T}\rvert}$
$P(X\lvert Y)$	$\frac{P(X,Y)}{P(Y)}$
$P(\neg X\lvert Y)$	$\frac{P(\neg X,Y)}{P(Y)}$
$P(X\lvert \neg Y)$	$\frac{P(X,\neg Y)}{P(\neg Y)}$
$P(\neg X\lvert \neg Y)$	$\frac{P(\neg X,\neg Y)}{P(\neg Y)}$

For example, *support* is the probability $P(A, B)$ (i.e., the probability that a transaction includes both A and B). Likewise, *confidence* is the probability $P(B|A)$ (i.e., the conditional probability that B is in a transaction given that A is there). In some more recent measures non-occurrence of the antecedent or consequent is also accounted for. For example, *casual support* is defined as $P(A, B) + P(\neg A, \neg B)$. Table 1 lists the probabilistic building blocks we consider and their definitions.

There is one final detail related to the interestingness measures that is relevant to our discussion: the *range* of a measure, and specifically its ability to measure either negative, positive, or no correlation between a rule's antecedent and consequent. We use the syntax $[min..mid..max]$ to give the range of an interestingness measures, *mid* here indicates the value of no correlation for that measure. When only $[min..max]$ is used, the value of no correlation is equal to 0.

The range of most measures falls into one of a few categories. Most existing measures (e.g., *support*) range between 0 and 1. This [0..1] range is also the easiest to interpret as a correlation, where 0 naturally indicates no correlation and any higher value the degree of positive correlation. Another common range is [-1..1], where 0 again indicates no correlation, but negative correlation is also possible. In addition, there also exist ranges such as $[-\infty..\infty]$ and $[0..1..\infty]$, where 1 indicates no correlation in the latter case. A complete list of the interestingness measures used in the study and their ranges is given in Table 2.

5. HYPER-RULES

To study the value of aggregating the evidence provided by a collection of conventional rules, we introduce the concept of a *hyper-rule*, which provides an effective summary of the constituent rules. A key requirement of hyper-rule formation is devising a method to properly aggregate the conventional rules' interestingness measures. This section first provides a few basic definitions. It then formalizes the notion of a

Table 2: Overview of the 39 interestingness measures considered in our study

#	Interestingness Measure	Range	Definition
1	Added Value	$[-0.5..1]$	$P(B\|A) - P(B)$
2	Casual Confidence	$[0..1]$	$\frac{1}{2} * (P(B\|A) + P(\neg B\|\neg A))$
3	Casual Support	$[0..1]$	$P(A,B) + P(\neg A, \neg B)$
4	Collective Strength	$[0..\infty]$	$\frac{P(A,b)+P(\neg B\|\neg A)}{P(A)*P(B)+P(\neg A)*P(\neg B)}$
5	Confidence	$[0..1]$	$P(B\|A)$
6	Conviction	$[0..1]$	$\frac{P(A)*P(\neg B)}{P(A,\neg B)}$
7	Cosine	$[0..1]$	$\frac{P(A,B)}{\sqrt{P(A)*P(B)}}$
8	Coverage	$[0..1]$	$P(A)$
9	Descriptive Confirmed Confidence	$[-1..1]$	$P(B\|A) - P(\neg B\|A)$
10	Difference Of Confidence	$[-1..1]$	$P(B\|A) - P(B\|\neg A)$
11	Example and Counterexample Rate	$[0..1]$	$(P(A,B) - P(A,\neg B))/P(A,B)$
12	Gini Index	$[0..1]$	$P(A) * (P(B\|A)^2 + P(\neg B\|A)^2) + P(\neg A) * (P(B\|\neg A)^2 + P(\neg B\|\neg A)^2) - P(B)^2 - P(\neg B)^2$
13	Imbalance Ratio	$[0..1]$	$\frac{\|P(A\|B)-P(B\|A)\|}{P(A\|B)+P(B\|A)-P(A\|B)*P(B\|A)}$
14	Interestingness Weighting Dependency (k=m=2)	$[0..\infty]$	$(\frac{P(B\|A)}{P(B)})^{(k-1)} * (P(A,B))^m$
15	J Measure	$[0..1]$	$P(A,B) * log(\frac{P(B\|A)}{P(B)}) + P(A,\neg B) * log(\frac{P(\neg B\|A}{P(\neg B}))$
16	Jaccard	$[-1..1]$	$\frac{P(A,B)}{(P(A)+P(B)-P(A,B))}$
17	Kappa	$[-1..1]$	$\frac{P(A,B)+P(\neg A,\neg B)-P(A)*P(B)-P(\neg A)*P(\neg B)}{1-P(A)*P(B)-P(\neg A)*P(\neg B)}$
18	Klosgen	$[-1..1]$	$\sqrt{P(A,B)} * (P(B\|A) - P(B))$
19	Kulczynski	$[0..1]$	$\frac{P(A,B)}{2} * (\frac{1}{P(A)} + \frac{1}{P(B)})$
20	Laplace Corrected Confidence	$[0..1]$	$\frac{P(A,B)+1}{P(B)+2}$
21	Least Contradiction	$[-1..1]$	$\frac{P(A,B)-P(A,\neg B)}{P(B)}$
22	Leverage	$[-1..1]$	$P(A,B) - P(A) * P(B)$
23	Lift	$[0..1..\infty]$	$\frac{P(A,B)}{P(A)} * P(B)$
24	Linear Correlation Coefficient	$[-1..1]$	$\frac{P(A,B)-P(A)*P(B)}{\sqrt{P(A)*P(B)*P(\neg A)*P(\neg B)}}$
25	Loevinger	$[-1..1]$	$\frac{P(B\|A)-P(B)}{1-P(B)}$
26	Odd Multiplier	$[0..\infty]$	$\frac{P(A,B)*P(\neg B)}{P(B)*P(A,\neg B)}$
27	Odds Ratio	$[0..1..\infty]$	$\frac{P(A,B)*P(\neg A,\neg B)}{P(A,\neg B)*P(\neg A,B)}$
28	One Way Support	$[0..\infty]$	$P(B\|A) * log_2(\frac{P(A,B)}{P(A)*P(B)})$
29	Prevalence	$[0..1]$	$P(B)$
30	Recall	$[0..1]$	$P(A\|B)$
31	Relative Risk	$[0..\infty]$	$\frac{P(B\|A)}{P(B\|\neg A)}$
32	Sebag Schoenauer	$[0..1]$	$\frac{P(A,B)}{P(A,\neg B)}$
33	Specificity	$[0..1]$	$P(\neg B\|\neg A)$
34	Support	$[0..1]$	$P(A,B)$
35	Two Way Support	$[0..\infty]$	$P(A,B) * log_2(\frac{P(A,B)}{P(A)*P(B)})$
36	Varying Rates Liaison	$[-1..\infty]$	$\frac{P(A,B)}{P(A)*P(B)} - 1$
37	Yules Q	$[-1..1]$	$\frac{\text{odds ratio}-1}{\text{odds ratio}+1}$
38	Yules Y	$[-1..1]$	$\frac{\sqrt{\text{odds ratio}}-1}{\sqrt{\text{odds ratio}}+1}$
39	Zhang	$[0..\infty]$	$\frac{P(A,B)-P(A)*P(B)}{max(P(AB)*P(\neg B),P(B)*P(A,\neg B))}$

hyper-rule and the process of measure aggregation.

5.1 Preliminary Definitions

The following definitions support the definition of a hyper-rule.

Definition 1 (Antecedent and Consequent) *Given a conventional rule, we define two functions: R^A and R^C, where the first returns the rule's antecedent and the second its consequent. Let $r = A \rightarrow B$ be a conventional rule. Then*

- $R^A(r) = A$
- $R^C(r) = B$

Definition 2 (Normalized Interestingness Measure) *An interestingness measure is* normalized *if (1) its range contains the value 0; and (2) the value 0 indicates no correlation between the rule and the change set.*

Most interestingness measures only specify positive correlation, with 0 indicating no correlation. However, some measures are defined with 1 indicating no correlation (where < 1 captures a negative correlation and > 1 a positive correlation). In order to formulate a consistent definition of interestingness measures aggregation, these measures must be normalized such that no correlation is indicated by 0. Out of all interestingness measures in Table 2, only *lift* and *odds ratio* are normalized.

Example 2 (Normalizing Around Zero) *The lift interestingness measure, has a range of $[0..1..\infty]$ with 1 indicating no correlation. To normalize the measure, we subtract 1 from each value, before aggregation. For example, we map the set of lift measures $\{0, 0.5, 10\}$, to the normalized set of measures: $\{-1, -0.5, 9\}$. The aggregated value is then converted back by adding 1.*

5.2 Hyper-Rule Formation Function

A hyper-rule, which intuitively summarizes a set of rules using a single rule, is formed using the following function:

Definition 3 (Hyper-Rule) *Given a set of rules*

$$R = \{r_1, \ldots, r_n\},$$

we define the hyper-rule formation function $\mathcal{H}(R)$ as

$$\mathcal{H}(R) = \bigcup_{r \in R} \{R^A(r)\} \quad \Rightarrow \quad \bigcup_{r \in R} \{R^C(r)\} \tag{8}$$

Note that the antecedent and the consequent of a hyper-rule are sets of sets of entities rather than being sets of entities as found in conventional rules. To help distinguish hyper-rules and conventional rules, we use a double-arrow $\Rightarrow$ in a hyper-rule rather than the single arrow $\rightarrow$ used with conventional rules.

Example 3 *Rule set $R = \{\{a, b\} \rightarrow \{c\}\}, \{\{d\} \rightarrow \{c\}\}\}$ will generate the hyper-rule*

$$\mathcal{H}(R) = \{\{a, b\}, \{d\}\} \Rightarrow \{\{c\}\}$$

Example 4 *Rule set $R = \{\{a, b\} \rightarrow \{c, f\}\}, \{\{a, b\} \rightarrow \{c\}\}\}$ will generate the hyper-rule*

$$\mathcal{H}(R) = \{\{a, b\}\} \Rightarrow \{\{c, f\}, \{c\}\}$$

Section 3 presented one possible rational for selecting a set of conventional rules to be combined – based on our experience with recommendation systems, it is advantageous to combine rules that have the same consequent. This is, of course, not the only possibility. For example, to answer the question "what does this file impact?" we would want to select for aggregation rules sharing the same antecedent. Beyond these two, other problem domains may require still other selection criteria.

5.3 Measure Aggregation

In the same manner that an association rule has an interestingness measure, a hyper-rule has an aggregated interestingness measure, which aggregates the individual interestingness measures of its constituent rules.

Let $\mathbb{M}$ denote the set of all normalized interestingness measures defined over conventional rules, and $\mathbb{H}$ denote the set of all hyper-rules. We define a *Unified Measure Aggregator* $\oplus : \mathbb{H} \times \mathbb{M} \rightarrow \mathbb{R}$ as follows:

Definition 4 (Unified Measure Aggregator) *Let M be a normalized interestingness measure defined over conventional rules, and $R = \{r_1, \ldots, r_n\}$ be a set of rules. Then $\oplus(\mathcal{H}(R), M)$ is a real number for which the following properties hold:*

1. $R = \{r\} \quad \Longrightarrow \quad \oplus(\mathcal{H}(R), M) = M(r)$
2. *If $|R| > 1$, then for each $r \in R$ the following holds:*

$$\oplus(\mathcal{H}(R), M) \begin{cases} > \oplus(\mathcal{H}(R - \{r\}), M) & M(r) > 0 \\ = \oplus(\mathcal{H}(R - \{r\}), M) & M(r) = 0 \\ < \oplus(\mathcal{H}(R - \{r\}), M) & M(r) < 0 \end{cases}$$

The first property provides identity by defining the aggregation of a single measure value as the value itself. This ensures that the hyper-rule transformation of a single rule, never returns a hyper-rule with a different measure value than the original rule. The second property ensures monotonicity. For example, it requires that the aggregation function is strictly increasing when rules with positive measures are aggregated, and strictly decreasing when rules with negative measures are aggregated.

6. AGGREGATION FUNCTIONS

Cumulative gain (CG) and discounted cumulative gain (DCG) are well known performance measures in information retrieval [12]. They are typically used to evaluate search results, and are generally used for evaluating a target list against an ideal (oracle) list [12].

In this paper we argue that CG, DCG, and a variation of DCG which we will refer to as DCG2 [6], are also well suited to the problem of aggregating interestingness values. Due to space restrictions, we omit formal proofs that the proposed aggregators satisfy the two properties introduced in Definition 4 in favor of presenting empirical evidence of their impact in Section 7.

We first informally introduce CG and DCG using the following example, set in their conventional context:

Example 5 *The result of a web search produced the following list of sites, prioritized from left to right: $L = [site_1, site_4, site_2]$. A user then ranks the search result by providing relevancy scores of each site on a scale of 0 to 3, producing the list:*

$$[(site_1, 1), (site_4, 3), (site_2, 2)]$$

CG and DCG can be applied to the list of relevancy scores [1,3,2], to provide an overall evaluation of the search result.

$$CG([1, 3, 2]) = 1 + 3 + 2 = 6$$
$$DCG([1, 3, 2]) = 1 + 3/log_2(2) + 2/log_2(3) \approx 5.26$$

In summary, CG assigns to L a score of 6 on a scale from 0 (all sites are not relevant) to 9 (all sites are highly relevant), indicating that, overall, the list is quite relevant. Note that CG ignores the order of the sites, and hence a similar list L where the most relevant site appears last would have the same score. In contrast, DCG uses a progressively decreasing weight, hence penalizing the fact that the first document returned is not marked by the user as very relevant.

6.1 Cumulative Gain

Definition 5 (Cumulative Gain) *Given a normalized interestingness measure M and a set of rules*

$$R = \{r_1, \ldots, r_n\},$$

the cumulative gain *of hyper-rule $\mathcal{H}(R)$ is defined as follows:*

$$\oplus_{CG}(\mathcal{H}(R), M) = \sum_{i=1}^{n} M(r_i) \tag{9}$$

Example 6 (Cumulative Gain) *Given the following set of rules R and their normalized interestingness measure M:*

$$R = \{r_1, r_2, r_3\}$$
$$M = \{(r_1, 1), (r_2, 3), (r_3, 2)\}$$

the cumulative gain *of $\mathcal{H}(R)$ is given by:*

$$\oplus_{CG}(\mathcal{H}(R)) = 1 + 3 + 2 = 6$$

Informally, as a simple algebraic sum, CG satisfies the aggregation properties of Definition 4.

6.2 Discounted Cumulative Gain

We introduce two discounted cumulative gain functions, referred to as DCG and DCG2 respectively. These functions use a coefficient to reduce the impact of lower ranked measures. In particular, when compared to DCG, DCG2 gives larger weight to highly ranked measures.

We begin by defining the notion of *absolute rank*, which is used in the definitions of DCG and DCG2.

Definition 6 (Absolute rank of a rule) *Let M be an interestingness measure, and $R = \{r_1, \ldots, r_n\}$ be a set of rules. Let $L = \langle r_{r_1}, ..., r_{r_n} \rangle$ be a ranking over R based on the absolute values of M for the rules in R (i.e., $|M(r_i)|$). The first element in L (i.e., r_{s_1}) is the rule for which M has the highest absolute value. Rules with the same absolute values for M get the same position in the rank.*

We define the absolute rank of a rule as its position in the ranking L.

In the following, for the sake of simplicity, we use $rank^+(r_i)$ to denote $rank^+(r_i, R, M)$.

Definition 7 (Discounted Cumulative Gain) *Given a normalized interestingness measure M, and a set of rules*

$$R = \{r_1, \ldots, r_n\},$$

DCG of $\mathcal{H}(R)$ for M is defined as follows:

$$\oplus_{DCG}(\mathcal{H}(R), M) = \sum_{i=1}^{n} M(r_i) \times coef(r_i) \tag{10}$$

where $coef(r_i)$ is given by:

$$coef(r_i) = \begin{cases} 1 & rank^+(r_i) = 1 \\ \frac{1}{log_2(rank^+(r_i))} & rank^+(r_i) > 1 \end{cases}$$

Example 7 (Discounted Cumulative Gain) *Given a set of rules R and a normalized interestingness measure M:*

$$R = \{r_1, r_2, r_3\}$$
$$M = \{(r_1, 1), (r_2, 3), (r_3, 2)\}$$

the DCG of $\mathcal{H}(R)$ for M is given by:

$$\oplus_{DCG}(\mathcal{H}(R), M) = 3 + 2/log_2(2) + 1/log_2(3) \approx 5.63$$

The next aggregator, DCG2, slightly modifies the function of DCG by assigning larger weight on highly positively or negatively correlated measures.

Definition 8 (Discounted Cumulative Gain 2 (Adapted)) *Given a normalized interestingness measure M, and a set of rules*

$$R = \{r_1, \ldots, r_n\},$$

DCG2 of $\mathcal{H}(R)$ for M is defined as follows:

$$\oplus_{DCG2}(\mathcal{H}(R), M) = \sum_{i=1}^{n} (2^{|M(r_i)|} - 1) \times coef(r_i) \tag{11}$$

where $coef(r_i)$ is defined as:

$$coef(r_i) = \begin{cases} 1 & rank^+(r_i) = 1 \\ \frac{sign(M(r_i))}{log_2(rank^+(r_i))} & rank^+(r_i) > 1 \end{cases}$$

Note that, in its original definition, DCG2 is only defined for positive measure values. We adapted the original definition of DCG2 in order to apply DCG2 to negative measure values too, and ensure that positive and negative values with the same absolute value have the same weight. Specifically, we adapt the definition of DCG2 to (1) consider in the exponent of the summation factor the absolute value of the value produced by interestingness measure M, and (2) defining *coef* such that it has the same sign of the interestingness measure.

7. EVALUATION

To assess the viability of hyper-rules, and especially the proposed aggregation functions of Section 6, we perform a large empirical study. The evaluation investigates the performance of hyper-rules with different aggregation functions, in the context of various software-systems and various interestingness measures.

While we believe hyper-rules will be useful in a variety of problem domains, our study focuses on change recommendation. Specifically, we investigate the following two research questions:

RQ 1 *How frequently are hyper-rules applicable in change recommendation?*

RQ 2 *Do hyper-rules positively impact the precision of change recommendation?*

The remainder of this section will give further detail on our evaluation setup, and is organized as follows: In Section 7.1 we explain the software-systems included in the study. In Section 7.2 we describe how the initial history is filtered. In Section 7.3 we describe two central concepts for the evaluation, namely *query creation* and *query execution*. In Section 7.4 we explain the generation of change recommendations. In Section 7.5 we discuss a slight simplification in the context of our evaluation. In Section 7.6 we explain our method for measuring performance, and in Section 7.7 we introduce the implementation and execution environment of the evaluation.

7.1 Subject Systems

To assess hyper-rules in a variety of conditions, we selected six systems having varying size and frequency of commits. Two of these systems come from our industry partners, Kongsberg Maritime (KM) and Cisco Norway. KM is a leading

company in the production of systems for positioning, surveying, navigation, and automation of merchant vessels and offshore installations. Specifically, we consider a common software platform KM uses in applications in the maritime and energy domain. Cisco Norway is the Norwegian division of Cisco Systems, a worldwide leader in the production of networking equipment. In particular, we evaluated hyper-rules on a software product line for professional video conferencing systems developed by Cisco Norway. The other four systems are parts of well known open-source projects, namely Apache HTTP Server, Linux Kernel, MySQL, and Git. Table 3 summarizes descriptive characteristics of the software systems used in the evaluation. The table shows that the systems we selected vary from medium to large size, with up to forty thousand different files committed in the transaction history. Furthermore, the oldest transactions from the system histories are fifteen years old in the case of KM. Note that all the systems are heterogeneous, i.e., they involve more than a one programming language.

7.2 History Filtering

In Table 3 we can see that commits with ten or fewer changed files are the most common. For this reason, and because we assume that larger commits often consist of unrelated files committed together because of a directory reshuffle or license change, we follow the work of Zimmerman et al. [31] and remove commits of more than thirty files from the original history. The resulting sequence of commits becomes the filtered history H_f, from which we will mine the association rules.

7.3 Query Creation

Conceptually, a *query* Q represents a set of files that a developer changed since the last synchronization with the version control system. The key idea behind our evaluation is to generate, starting from a transaction T, a set of queries that emulate a developer errantly forgetting to update some subset of T.

From the filtered history H_f, we sample fifty commits of each size between two and ten. By sampling evenly over this range, rather than sampling all available commits of each size, we avoid a bias towards the smallest commits of size 2-3, which are far more common [23]. The resulting 450 commits are used to form queries that are executed in the various contexts we want to investigate.

To mimic a developer forgetting to change one or more files, we partition each transaction T into a non-empty query Q and a non-empty expected outcome $E \stackrel{\text{def}}{=} T \setminus Q$. In this way, we can evaluate the ability of a recommendation to infer E from Q.

To investigate a range of query sizes, we generate one query for each size from 1 to $|T| - 1$. For example, for a transaction of size 4, we generate three queries of sizes 1, 2, and 3, whose expected outcomes thus have sizes 3, 2 and 1, respectively. Note that we do not sample commits of size one because they cannot be split into a query and expected outcome.

7.4 Generating Change Recommendations

All queries are executed using two different targeted association rule mining algorithms, namely TARMAQ and CO-CHANGE (presented in Section 2). Executing a query Q, created from a transaction T, creates a set of association rules. The rules may differ based on which algorithm that was used. Moving from a set of rules to a change-recommendation with respect to Q, requires giving weight to the rules such that they can be sorted. In this paper we consider all the of the interestingness measures of Table 2 for this purpose. Moreover, and key to the purpose of this paper, we also convert the set of weighted rules into their aggregated versions, according to the hyper-rule definition of Section 5 and the aggregation functions of Section 6.

The rules, and interestingness measures used to weight rules, are based on patterns in previous commits, and for this purpose we use the 10,000 commits made prior to T. Note that this means that all queries are executed over their *actual* previous history. The use of 10,000 commits represents a balance between to short a history, which would lack sufficient connections, and to long a history, which is inefficient and can even be misleading when previously connected files are no longer connected.

7.5 Aggregation of only Positive Measures

As discussed in Section 4, interestingness measures typically capture either only positive, or both positive and negative correlations between the antecedent and consequent of a rule. In our evaluation, we decided to consider only positive correlations. This means that for the interestingness measures that can produce both positive and negative correlations, we ignore any rule which has a negative measure value. This is in line with our problem domain, i.e., "if I change 'this', what else should be changed?", rather than: "if I change 'this', what else should not be changed?".

Moreover, existing targeted association rule mining algorithms show a clear bias toward mining only positive rules (typically based on entities that have changed together in the past). Thus the interestingness measures that have the ability to measure both negative and positive correlations will be heavily skewed towards the positive correlations.

Looking forward, there is potential value in combining the two types of correlation. Investigating such is left to future work.

7.6 Performance Measure

To evaluate a recommendation we use the *average precision* (AP) measure:

Definition 9 (Average Precision) *Given a recommendation* R*, and an expected outcome* E*, the average precision of* R *is given by:*

$$AP(R) \stackrel{def}{=} \sum_{k=1}^{|R|} P(k) * \triangle r(k) \tag{12}$$

where $P(k)$ is the precision calculated on the first k files in the list (i.e., the fraction of correct files in the top k files), and $\triangle r(k)$ is the change in recall *calculated only on the $k-1^{th}$ and k^{th} files (i.e., how many more correct files where predicted compared to the previous rank).*

Note that since we consider only rules with single consequents, $\triangle r(k)$ will always be equal to either zero or $1/|E||$, i.e., a rank either does not contain a file from the expected outcome, or it contains exactly one file from the expected outcome. Table 4 illustrates the computation of AP, $P(k)$, and $\triangle r(k)$ given the ranked list $[c, a, f, g, d]$ and the expected outcome $\{c, d, f\}$.

As an overall performance measure of a group of factors, e.g., a certain rule generation algorithm using a certain interestingness measure, we use the *mean average precision*

Table 3: Characteristics of the evaluated software systems

Software System	Unique files	Avg. transaction size (# files)	History covered by 10 000 transactions	Languages used*
MySQL	21854	10.1	2.34 years	C++ (54%), C (19%), JavaScript (17%), 23 other (10%)
Git	2141	1.9	4.2 years	C (45%), shell script (35%), Perl (9%), 14 other (11%)
Apache HTTP Server	7905	6.9	7.18 years	XML (56%), C (32%), Forth (8%), 19 other (4%)
Linux Kernel	9021	2.2	0.15 years	C (94%), 16 other (6%)
Kongsberg Maritime	35111	5.1	15.97 years	C++, C, XML, other build/config (% undisclosed)
Cisco Norway	41701	6.2	1.07 years	C++, C, C#, Python, Java, XML, other build/config (% undisclosed)

* data on the languages used by the open source systems was obtained from https://www.openhub.net/

(MAP) computed over all the queries executed using that factor combination.

7.7 Experimental Setup

All rule generation algorithms, interestingness measures, and hyper-rules have been implemented in Ruby. We performed the experiment on a *m4.2xlarge* Amazon EC2 instance.[2]

8. RESULTS

This section presents the results of the study described in Section 7. We discuss the implications of these results in Section 9. The results are broken into two parts. First, in Section 8.1, we consider RQ1, the applicability of hyper-rule formation and then, in Section 8.2, we turn to RQ2 and the impact of hyper-rule formation.

8.1 Applicability of Hyper-Rules (RQ 1)

Even if a technique has been proven to increase performance, it can be of little practical value if the prerequisites for its application are too restrictive. As such, it is of great interest to gain insight into how often hyper-rules are applicable in the change recommendation context.

To investigate this aspect, we calculated the percentage of recommendations having at least two rules with the same consequent. In answer to RQ1, the percentages are given in Table 5. For Co-Change 70% of all recommendations have at least two rules from which a hyper-rule can be formed. The percentage for Tarmaq, 15%, is notably lower, but still provides opportunities for hyper-rule formation.

8.2 Ability to Improve Precision (RQ 2)

The results for RQ1 indicate that there exist opportunities to apply hyper-rule formation. To address RQ2, we evaluate the benefit that their formation brings. The evaluation considers three aspects: the rule generation algorithm used (Co-Change or Tarmaq), the interestingness measure used to rank the rules, and the aggregation function used to form the hyper-rules.

The investigation considers the impact of hyper-rule formation on the average precision attained by a recommendation. In doing so, we also identify the best performing aggregator(s) for each of the algorithm and interestingness-measure combination. Hereafter we refer to an (algorithm, measure) combination as a *case*.

The overall results are shown in Table 6. Each column shows the percentage increase or decrease in mean average precision for the three different aggregators, compared to no aggregation.

8.2.1 Cases where hyper-rules change precision

To test if aggregation has a significant impact on the recommendations, we use the Friedman Test on each case. This test was used because the distributions are not necessarily normal. In Table 6, we highlight with a dashed box the case (Tarmaq, *kappa*) that produced the largest p-value, of 0.021. In all other cases, we found a highly significant difference between aggregation and no aggregation. Note that the Friedman test does not indicate that aggregation has a positive effect, only that it has *an effect*.

8.2.2 Best performing measure aggregators

For each significant difference, a post-hoc test is used to determine the aggregators that caused the difference. For this purpose, we use multiple one-tailed, paired Wilcoxon signed rank tests with the Bonferroni adjustment. The adjustment accounts for the multiple comparisons by lowering the significance threshold for the p-values, in order to address potential false positive resulting from multiple tests. To obtain a partial ordering of the aggregators, we first run three tests that compare the aggregators against no aggregation. Thereafter, we perform three additional tests between the aggregators, for a total of 6 tests (comparisons). Note that the Bonferroni adjustment is α divided by the number of comparisons, which gives us an adjusted α of $0.05/6 \approx 0.008$,

The results of the Wilcoxon tests are shown in Table 6, where bold and underlined values indicate the aggregator

[2] https://aws.amazon.com

Table 4: Example of average precision calculation

Consider as relevant files: c, d, f

Rank (k)	File	$P(k)$	$\triangle r(k)$
1	c	1/1	1/3
2	a	1/2	0
3	f	2/3	1/3
4	g	2/4	0
5	d	3/5	1/3

$AP = 1/1 * 1/3 + 1/2 * 0 + 2/3 * 1/3 + 2/4 * 0 + 3/5 * 1/3 \approx 0.75$

Table 5: Percentage of recommendations where at least two rules could be aggregated into a hyper-rule

	Applicable	Non Applicable
Co-Change	70%	30%
Tarmaq	15%	85%

Table 6: The change in Mean Average Precision for the various aggregator functions compared to recommendations without aggregation.

	Added Value	Casual Confidence	Casual Support	Collective Strength	Confidence	Conviction	Cosine	Coverage	Descriptive Confirmed confidence	Difference of Confidence	Example and Counterexample Rate	Gini Index	Imbalance Ratio	Interestingness Weighting Dependency	J-Measure	Jaccard	Kappa	Klosgen	Kulczynski	Laplace Corrected Confidence	Least Contradiction	Leverage	Lift	Linear Correlation Coefficient	Loevinger	Odd Multiplier	Odds Ratio	One Way Support	Prevalence	Recall	Relative Risk	Sebag Schoenauer	Specificity	Support	Two Way Support	Varying Rates Liaison	Yules Q	Yules Y	Zhang
Co-Change algorithm																																							
CG	**7%**	2%	**26%**	**24%**	**7%**	3%	**14%**	**83%**	-2%	**7%**	-2%	**32%**	**53%**	3%	**80%**	**11%**	**11%**	5%	**9%**	**26%**	0%	**14%**	**19%**	**14%**	**7%**	**8%**	4%	**142%**	**18%**	**38%**	**8%**	3%	**100%**	**14%**	**53%**	**19%**	**7%**	**11%**	**9%**
DCG	**7%**	5%	**26%**	**24%**	**7%**	3%	**11%**	**83%**	-2%	**7%**	-2%	**32%**	**53%**	3%	**80%**	**8%**	**11%**	2%	**9%**	**30%**	0%	**11%**	**14%**	**11%**	**7%**	**8%**	4%	**142%**	**18%**	**31%**	**8%**	5%	**100%**	**14%**	**53%**	**14%**	**7%**	**11%**	**9%**
DCG2	5%	-9%	-19%	-24%	5%	-3%	0%	17%	-5%	5%	-5%	**32%**	13%	0%	**80%**	3%	0%	0%	-12%	-19%	-2%	3%	5%	0%	2%	-8%	-11%	**142%**	5%	13%	-4%	3%	27%	3%	**53%**	5%	-19%	-15%	-32%
Tarmaq algorithm																																						
CG	**35%**	**19%**	**35%**	7%	**19%**	9%	**26%**	**37%**	**15%**	**35%**	**15%**	**23%**	**24%**	0%	**56%**	5%	7%	**19%**	**47%**	**67%**	4%	**24%**	7%	**21%**	**19%**	12%	11%	**56%**	**9%**	8%	6%	**9%**	**56%**	**11%**	**24%**	7%	**17%**	**17%**	**17%**
DCG	**35%**	**19%**	**35%**	7%	**19%**	9%	**21%**	**37%**	**15%**	**35%**	**15%**	**23%**	**24%**	0%	**56%**	5%	7%	**19%**	**47%**	**67%**	4%	**24%**	7%	**21%**	**19%**	12%	11%	**56%**	**9%**	8%	6%	9%	**56%**	**11%**	**24%**	7%	**17%**	**17%**	**17%**
DCG2	**35%**	15%	**30%**	3%	**15%**	9%	16%	**32%**	**15%**	**35%**	**15%**	**23%**	19%	0%	**56%**	5%	7%	**19%**	**37%**	**60%**	0%	**20%**	**67%**	**16%**	**15%**	**18%**	**17%**	**56%**	9%	0%	**47%**	**9%**	**56%**	**7%**	**24%**	**67%**	**17%**	**17%**	**17%**

Bold and underlined values indicate aggregators that performed significantly better than no aggregation*, as well as the other (non-bold) aggregators for that interestingness measure and algorithm. The three values in the marked rectangle are the only ones for which there was close to no statistical difference between the aggregators according to a Friedman test (p-value of 0.021 with $\alpha = 0.05$).*

that performed significantly better than the others for each interestingness measure. In the cases where there are $N > 1$ bold values, there is no significant difference between the top N aggregators. For example, in the case (Tarmaq, casual confidence), CG and DCG are distinctively better than no aggregation and the DCG2 aggregator. In another case (Co-Change, J-measure) CG, DCG, DCG2 were all found to be significantly better than no aggregation.

As seen in Table 6 by the substantial presence of bolded values, hyper-rule formation has had an overall positive impact on recommendation performance.

9. DISCUSSION

This section discusses the implications of the results presented in Section 8.

9.1 Applicability of Hyper-Rules (RQ 1)

In Section 8.1 we found that Co-Change generated recommendations from which hyper-rules could be formed 70% of the time while Tarmaq does so only 15% of the time. However, this relatively large difference should not come as a surprise. In the problem domain of change recommendations, a rule-set is convertible to a hyper-rule if all the rules shared the same consequent. Thus the nature of the algorithm that generates the rule-set affects the applicability of hyper-rule formation.

In the case of Co-Change, only rules with a single antecedent and a single consequent are generated, while Tarmaq generates rules where the antecedent is potentially equal to the change-set. Thus the number of rules generated by Co-Change is potentially much larger than the number generated by Tarmaq. With this in mind, the percentages in Table 5 provide a positive outlook for hyper-rule formation. We conclude this because Co-Change and Tarmaq likely represent two ends of a spectrum, making, the percentage of applicable recommendations for other algorithms likely to lie in the range 15% – 70%.

9.2 Ability to Improve Precision (RQ 2)

From the results presented in Table 6, we can observe the following for our two studied algorithms:

- For Co-Change, DCG2 stands out as overall having less, and sometimes even negative impact. On the other hand, CG and DCG improve the recommendations for 30 of 39 interestingness measures.
- For Tarmaq, all aggregators consistently perform better across most interestingness measures compared to Co-Change. There is also less difference between the aggregators. In a few cases however, DCG2 performs much better than CG and DCG. Aggregation improved recommendations for 32 of 39 interestingness measures.

The reason behind these differences comes back to the nature of the two algorithms. For example, in the case of Tarmaq, we see less difference between aggregators as a result of having fewer rules to aggregate.

The differences also bring out a more subtle behavior of the DCG2 aggregator. DCG2 is a fast growing function for larger measures. It thus can have a large impact on ordering in recommendations even when there are only a few rules. From Table 6 we see that this has a positive effect when used with Tarmaq, and an adverse effect when used with Co-Change. This difference is evident with the measures Lift, Relative Risk, and Varying Rates Liaison. All of these measures have a range whose right bound is ∞, and are thus much more likely to grow fast when DCG2 is used as an aggregator. We see that this faster growth has a positive effect in the "short term", i.e., aggregation of only a few rules.

To summarize, in general hyper-rule formation has a positive effect on change recommendation. However, the rule generation algorithm and interestingness measure effect the outcome, which is not surprising. In this paper, we have suggested and evaluated three aggregators, but it is potentially beneficial to investigate a wider range of aggregators for each measure. In particular, for the interestingness measures for which we found no significant recommendation improvement when aggregated, other aggregators should be investigated.

9.3 Threats to validity

Problem Domain used in Evaluation: We evaluated hyper-rules in the context of change recommendations. However, the different interestingness measures studied might not fit well into all problem domains [25]. Still, since we evaluated hyper-rules by looking at the *difference in precision* compared to not using hyper-rules, rather than looking

at the actual precision, we believe that the effect of the problem domain is minimized.

Implementation: We implemented and thoroughly tested all algorithms, aggregators and interestingness measures studied in this paper in Ruby. However, we can not guarantee the absence of implementation errors which may have affected our evaluation.

Variation in software systems: We evaluated hyper-rules on two industrial systems and four large open source systems. These systems vary considerably in size and frequency of transactions (commits), which should provide an accurate picture of the performance of hyper-rules in various contexts. However, despite our careful choice, we are likely not to have captured all variations.

Commits as basis for evolutionary coupling: The evaluation of this paper is grounded in predictions made from analysing patterns in change-histories. The transactions that make up the change-histories are however not in any way guaranteed to be "correct" or "complete", in the sense that they represent a coherent unit of work. Non-related files may be present in the transactions, and related files may be missing from the transactions. However, the included software-systems in our evaluation all (except KM) use Git for version control. As Git provides developers with tools for history-rewriting, we do believe that this might cause more coherent transactions.

10. RELATED WORK

We distinguish related work on aggregating association rules, clustering and pruning association rules, and on comparing interestingness measures.

Aggregating Association Rules: To the best of our knowledge, no other work has investigated the aggregation of association rules with the goal of combining the evidence (or interestingness) provided by individual rules.

Massoud et al. address the challenge of mining multi-dimensional association rules that aim to combine and relate association rules generated from two or more different sets of transactions [19]. They do not aggregate rules that combine evidence for the same conclusion but aim to create aggregate rules that span the dimensions of all transactions.

Clustering and Pruning Association Rules: Several authors investigate methods to discover the most informative or useful rules in a large collection of mined association rules, for example by clustering rules that convey redundant information, or by pruning *non-interesting* rules. Thus, while our method aims to aggregate rules to combine all existing evidence, this work tries to keep (or only generate) the "most important" rules. Toivonen et al. present *association rule covers* as a method to reduce the number of redundant rules [26]. Their method first groups rules which shared the same consequent, and then filters this set by considering the size of the antecedent in combination with the interestingness measures of the rules. No association rules or interestingness measures are aggregated. Kannan and Bhaskaran empirically study how such rule clusters are distributed over different interestingness measures [15]. Zaki introduces the *closed* frequent itemset as an alternative association rule mining technique that only generate non-redundant association rules [29]. The number of redundant rules produced by the new approach is dramatically smaller than the rule set from the traditional approach but this is achieved at generation time, i.e., no association rules or interestingness measures are aggregated. Baralis et al. investigate an association rule mining technique that combines *schema constraints* (i.e., rule constraints) and rule taxonomies to filter out redundant rules [3]. As with Zaki's approach, this is achieved at generation time, and no association rules or interestingness measures are aggregated. Liu et al. introduce *direction setting rules* as a method of summarizing the set of rules for a human user [17]. Essentially, direction setting rules are simple rules which capture part of the same relationships also captured in larger rules, i.e., they are more concise.

Selecting and Comparing Interestingness Measures: In Table 6, we provide empirical evidence that shows which aggregator performs best with each interestingness measure. Several authors have investigated properties of interestingness measures, and addressed how selecting the right measure for the problem domain in question can affect recommendation accuracy [25, 16, 18, 10]. To apply hyper-rules in a given problem domain we advise that these works be consulted, to help select an appropriate measure. Then Table 6 can be used to choose the best aggregator for the selected measure and algorithm.

11. CONCLUDING REMARKS

This paper introduces a novel approach that aggregates mined association rules forming *hyper-rules* that combine the evidence brought by each of the constituent rules. We evaluate the viability of hyper-rule formation in the context of change recommendation using a large empirical study of four open source systems as well as two systems from our industry partners. Based on this study, this paper makes the following contributions: (1) We identify an opportunity, missed by traditional recommendation systems, to increase accuracy using the increased evidence that is indicated by having multiple applicable rules in support of a particular conclusion. (2) We provide a theoretical foundation for rule aggregation using hyper-rules, and present three aggregation strategies for their interestingness measures (each rooted in existing work from information retrieval), (3) We empirically investigate how frequent rule aggregation can be applied in practice for two different association rule mining algorithms. (4) We provide empirical evidence that aggregation increases precision in 30 of the 39 interestingness measures when using Co-Change, and 32 of the 39 when using Tarmaq. (5) Results from our empirical study, can be used to select the best aggregator for use with each interestingness measure.

Directions for Future Work: In the future we would like to address the following: (1) Investigate the behavior of hyper-rules with additional algorithms in the change-recommendation domain. (2) Explore the use of hyper-rules in other domains. (3) Define interestingness measures directly on hyper-rules, rather than calculating aggregated measures. (4) Applying pre-filtering techniques such as those presented in Section 10, which may further improve the viability of hyper-rules.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. "Mining association rules between sets of items in large databases". In: *ACM SIGMOD International Conference on Management of Data*. ACM, 1993, pp. 207–216.

[2] T. Ball, J. Kim, and H. P. Siy. "If your version control system could talk". In: *ICSE Workshop on Process Modelling and Empirical Studies of Software Engineering*. 1997.

[3] E. Baralis et al. "Generalized association rule mining with constraints". In: *Information Sciences* 194 (2012), pp. 68–84.

[4] D. Beyer and A. Noack. "Clustering Software Artifacts Based on Frequent Common Changes". In: *13th International Workshop on Program Comprehension (IWPC)*. IEEE, 2005, pp. 259–268.

[5] S. Bohner and R. Arnold. *Software Change Impact Analysis*. CA, USA: IEEE, 1996.

[6] C. Burges et al. "Learning to rank using gradient descent". In: *Proceedings of the 22nd international conference on Machine learning - ICML '05*. Vol. pages. New York, New York, USA: ACM Press, 2005, pp. 89–96.

[7] G. Canfora and L. Cerulo. "Impact Analysis by Mining Software and Change Request Repositories". In: *11th IEEE International Software Metrics Symposium (METRICS)*. IEEE, 2005, pp. 29–37.

[8] S. Eick et al. "Does code decay? Assessing the evidence from change management data". In: *IEEE Transactions on Software Engineering* 27.1 (2001), pp. 1–12.

[9] H. Gall, K. Hajek, and M. Jazayeri. "Detection of logical coupling based on product release history". In: *IEEE International Conference on Software Maintenance (ICSM)*. IEEE, 1998, pp. 190–198.

[10] L. Geng and H. J. Hamilton. "Interestingness measures for data mining". In: *ACM Computing Surveys* 38.3 (Sept. 2006).

[11] A. Hassan and R. Holt. "Predicting change propagation in software systems". In: *20th IEEE International Conference on Software Maintenance, 2004. Proceedings*. IEEE, 2004, pp. 284–293.

[12] K. Järvelin and J. Kekäläinen. "Cumulated gain-based evaluation of IR techniques". In: *ACM Transactions on Information Systems* 20.4 (Oct. 2002), pp. 422–446.

[13] M.-A. Jashki, R. Zafarani, and E. Bagheri. "Towards a more efficient static software change impact analysis method". In: *8th ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering (PASTE)*. ACM, 2008, pp. 84–90.

[14] R. Kamber, Micheline and Shinghal. "Evaluating the Interestingness of Characteristic Rules". In: *KDD*. 1996, pp. 263–266.

[15] S. Kannan and R. Bhaskaran. "Association Rule Pruning based on Interestingness Measures with Clustering". In: *Journal of Computer Science* 6.1 (Dec. 2009), pp. 35–43.

[16] T.-d. B. Le and D. Lo. "Beyond support and confidence: Exploring interestingness measures for rule-based specification mining". In: *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. IEEE, Mar. 2015, pp. 331–340.

[17] B. Liu, W. Hsu, and Y. Ma. "Pruning and summarizing the discovered associations". In: *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '99* (1999), pp. 125–134.

[18] K. Mcgarry. "A survey of interestingness measures for knowledge discovery". In: *The Knowledge Engineering Review* 20.01 (2005), p. 39.

[19] R. B. Messaoud et al. "Enhanced mining of association rules from data cubes". In: *Proceedings of the 9th ACM international workshop on Data warehousing and OLAP - DOLAP '06*. New York, New York, USA: ACM Press, 2006, p. 11.

[20] A. Podgurski and L. Clarke. "A formal model of program dependences and its implications for software testing, debugging, and maintenance". In: *IEEE Transactions on Software Engineering* 16.9 (1990), pp. 965–979.

[21] X. Ren et al. "Chianti: a tool for change impact analysis of java programs". In: *ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications (OOPSLA)*. 2004, pp. 432–448.

[22] R. Robbes, D. Pollet, and M. Lanza. "Logical Coupling Based on Fine-Grained Change Information". In: *Working Conference on Reverse Engineering (WCRE)*. IEEE, 2008, pp. 42–46.

[23] T. Rolfsnes et al. "Generalizing the Analysis of Evolutionary Coupling for Software Change Impact Analysis". In: *23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. 2016, p. 12.

[24] R. Srikant, Q. Vu, and R. Agrawal. "Mining Association Rules with Item Constraints". In: *International Conference on Knowledge Discovery and Data Mining (KDD)*. AASI, 1997, pp. 67–73.

[25] P.-N. Tan, V. Kumar, and J. Srivastava. "Selecting the right objective measure for association analysis". In: *Information Systems* 29.4 (June 2004), pp. 293–313.

[26] H. Toivonen et al. "Pruning and Grouping Discovered Association Rules". In: *Workshop on Statistics, Machine Learning, and Knowledge Discovery in Databases*. Heraklion, Crete, Greece, 1995, pp. 47–52.

[27] A. R. Yazdanshenas and L. Moonen. "Crossing the boundaries while analyzing heterogeneous component-based software systems". In: *IEEE International Conference on Software Maintenance (ICSM)*. ICSM '11. Washington, DC, USA: IEEE, Sept. 2011, pp. 193–202.

[28] A. Ying et al. "Predicting source code changes by mining change history". In: *IEEE Transactions on Software Engineering* 30.9 (2004), pp. 574–586.

[29] M. J. Zaki. "Generating non-redundant association rules". In: *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '00*. New York, New York, USA: ACM Press, 2000, pp. 34–43.

[30] M. B. Zanjani, G. Swartzendruber, and H. Kagdi. “Impact analysis of change requests on source code based on interaction and commit histories”. In: *Working Conference on Mining Software Repositories (MSR)* (2014), pp. 162–171.

[31] T. Zimmermann et al. “Mining version histories to guide software changes”. In: *IEEE Transactions on Software Engineering* 31.6 (2005), pp. 429–445.