Evaluation of Tools for Hairy Requirements and Software Engineering Tasks: a correction

Daniel M. Berry
Cheriton School of Computer Science
University of Waterloo
Waterloo, ON, Canada
dberry@uwaterloo.ca

This copy and update of a published workshop paper is to correct an error in the context of a venue that does not allow publishing errata. The goal is to make any search that finds the original paper find this correction. This update includes an annotated copy of the original, with the error marked on Page 6 of the annotated copy, Page 7 of this document, and a correction which is a page appended to the end.

See page 6 for the description of an error that is corrected on the additional last page.

Evaluation of Tools for Hairy Requirements and Software Engineering Tasks

Daniel M. Berry
Cheriton School of Computer Science
University of Waterloo
Waterloo, ON, Canada
dberry@uwaterloo.ca

***Abstract*—*[Context and Motivation]* A *hairy* requirements or software engineering *task* involving natural language (NL) documents is one that is not inherently difficult for NL-understanding humans on a small scale but becomes unmanageable in the large scale. A hairy task demands tool assistance. Because humans need help in carrying out a hairy task completely, a tool for a hairy task should have as close to 100% recall as possible. A hairy task tool that falls short of close to 100% recall that is applied to the development of a high-dependability system may even be useless, because to find the missing information, a human has to do the entire task manually anyway. For a such a tool to have recall acceptably close to 100%, a human working with the tool on the task must achieve better recall than a human working on the task entirely manually. *[Problem]* Traditionally, many hairy requirements and software engineering tools have been evaluated mainly by how high their precision is, possibly leading to incorrect conclusions about how effective they are. *[Principal Ideas]* This paper describes using recall, a properly weighted *F*-measure, and a new measure called *summarization* to evaluate tools for hairy requirements and software engineering tasks and applies some of these measures to several tools reported in the literature. *[Contribution]* The finding is that some of these tools are actually better than they were thought to be when they were evaluated using mainly precision or an unweighted *F*-measure.**

***Index Terms*—abstraction finding, ambiguity finding, *F*-measure, false negatives, false positives, hairy task, manual task, natural language documents, precision, recall, summarization, tool-assisted task, tracing**

I. Introduction

A *hairy*[1] requirements or software engineering (RE or SE) *task* involving natural language (NL) documents is one that requires NL understanding [1] and is not difficult for humans to do on a small scale but is unmanageable when it is done to the documents that accompany the development of a large computer-based system (CBS) [2]. Any hairy task mentioned in the rest of this paper is assumed to be done in and for the development of a large CBS. Examples of hairy tasks include: finding abstractions and similar artifacts [3]–[8], finding trace links [9]–[13], consolidating multiple requirements specifications [14], merging multiple versions of a module [15], classifying requirements [16], analyzing app reviews [17], [18], synthesizing models from NL text [19], finding ambiguities in a requirements specification [20]–[24], and its generalization, finding defects in a requirements specification [25]–[27]. Such a hairy task is burdensome enough that tool assistance is needed to help a human to do complete job [28].

Humans understand NL well enough that a human has the potential of achieving for the hairy task close to 100% *correctness*, i.e., finding close to all and only the desired information. The two components of "correctness" are *recall*, that all the desired information is found, and *precision*, that only the desired information is found. Of these two components, for a hairy task, recall is more in need of tool assistance. For any task for which tool assistance is truly needed, finding a unit of desired information among the many documents available for the CBS's development is generally significantly harder than dismissing a found unit of information that is not desired. It is like finding needles in a haystack when one does not know how many needles the haystack has. If recall were not the harder component of correctness, we would not be building a tool to search for correct information. Therefore, for a hairy task, if close to 100% correctness is needed, then close to 100% recall is needed.

Not every instance of a hairy task needs close to 100% recall, and it may be enough for a tool for it to achieve only some recall. For example, if the task is to determine from NL app reviews whether a particular app has been actively used, finding any review that describes the app as being used suffices [17]. In such a case, it would be more important for the tool to achieve high precision, so that the user does not have to wade through many nonsense answers to find the correct answers that are buried in the output. The rest of this paper is about hairy tasks that require close to 100% recall.

Even for a hairy task that often requires close to 100% recall, not every application of it during the development of a CBS needs to achieve close to 100% recall. However, if the CBS being developed has high-dependability (HD) requirements, then recall for the task must be as close as possible to 100% in order to ensure that the needed dependability will be achieved [28]. For example, 100% of all trace links must be found in order to *ensure* that all the effects of any proposed change can be traced [29]. *In this HD circumstance*, if a tool for the task achieves less than close to 100% recall, then the task must be done manually on the whole of the documents to find the answers that the tool does not deliver. Therefore, in the last analysis, *in this HD circumstance*, such a tool is

[1]This author chose the word "hairy" to evoke the metaphor of the hairy theorem or proof.

really useless[2].

Just how close to 100% must the recall of a tool for a hairy task be? First, recognize that

- achieving 100% recall is probably impossible, even for a human, as is finding all bugs in a program, particularly because the task is hairy, and
- we have no way to know if a tool *has* achieved 100% recall, because the only way to measure recall for a tool is to compare the tool's output against the set of all correct answers, which is impossible to obtain, even by humans.

Let us call what humans can achieve when performing the task manually under the best of conditions the "humanly achievable high recall (HAHR)" for the task, which we hope is close to 100%. If a tool can be demonstrated to achieve better recall than the HAHR for its task, then a human will trust the tool and will not feel compelled to do the tool's task manually to look for what the human feels that the tool failed to find.

Thus, the real goal for any tool for a hairy task is to achieve the HAHR for the task. Therefore, a tool for a hairy task must be evaluated by empirically comparing the recall of humans working with the tool to carry out the task with the recall of humans carrying out the task manually [3], [8], [30]. Empirical studies will be needed to estimate the HAHR and other key values that inform the evaluations. See Section VI for a description of how this study can be performed as part of the construction of the gold standard for evaluation of the tools for a task.

The rest of this paper (II) defines recall (R), precision (P), and the F-measure, (III) describes when R should weighted more than P, (IV) discusses other sightings of this weighting idea, (V) defines F_β that allows weighting R or P over the other, (VI) summarizes an empirical method to evaluate hairy tools, (VII) introduces *summarization*, (VIII) categorizes related tool evaluation work by how its tools are evaluated, (IX) applies the empirical method to redo one categorized evaluation, and (X) considers human factors in evaluations.

This paper is a trimming of a longer technical report [31], hereinafter called "the full paper", which contains material left out of this paper to allow it to meet its page limitation.

II. Traditional Measures to Evaluate Tools

The traditional measures by which a tool for a hairy task, and for that matter any NLP-based tool, is evaluated are [11], [30]

- *recall*, the percentage of the correct answers that the tool finds, and
- *precision*, the percentage of the tool-found answers that are correct.

To define these measures precisely, it is necessary to consider in Figure 1 the universe of a tool for a hairy task. The

[2]Of course, one can argue that such a tool is useful as a defense against a human's less-than-100% recall when the tool is run as a double check after the human has done the tool's task manually. However, it seems to this author, that if the human *knows* that the tool will be run, he or she might be lazy in carrying out the manual task and not do as well as possible. Empirical studies are needed to see if this effect is is real, and if so, how destructive it is of the human's recall.

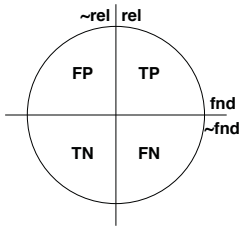

Fig. 1. The Universe of a Tool for a Hairy Task

circle represents the space of all possible answers, relevant, i.e., correct, or not, for the task that any tool might find or not. The space can be partitioned by two independent axes,

1) one separating the answers that the tool finds, fnd, from those that the tool does not find, $\sim$ fnd, and
2) one separating the answers that are relevant for the task, rel, from those that are not relevant, $\sim$ rel.

These two partitions create four regions in the space, consisting of the

- true positives, TP, that are the relevant answers that are found,
- false negatives, FN, that are the relevant answers that are not found,
- true negatives, TN, that are the not relevant answers that are not found,
- false positives, FP, that are the not relevant answers that are found

by the tool.

With these subspaces, it is possible to give precise definitions of recall, R, and precision, P:

$$R = \frac{|\mathsf{fnd} \cap \mathsf{rel}|}{|\mathsf{rel}|} = \frac{|\mathrm{TP}|}{|\mathrm{TP}| + |\mathrm{FN}|} \tag{1}$$

$$P = \frac{|\mathsf{fnd} \cap \mathsf{rel}|}{|\mathsf{fnd}|} = \frac{|\mathrm{TP}|}{|\mathrm{FP}| + |\mathrm{TP}|} \tag{2}$$

The composite of recall and precision that is often called "correctness" is captured by the F-measure:

$$F = 2 \times \frac{P \times R}{P + R} \quad , \tag{3}$$

the harmonic mean of recall, R, and precision, P.

III. Recall vs. Precision

For a typical hairy task, manually finding a missing correct answer, a false negative, is significantly harder than rejecting as nonsense an incorrect answer, a false positive[3], because finding a missing correct answer generally requires examining all the input documents in detail, while rejecting an incorrect answer generally requires understanding only the incorrect answer and the input documents at only a general level. For

[3]It seems reasonable to even include in the definition of a hairy task the proviso that manually finding a true positive or false negative is significantly harder than rejecting a proposed answer that is a false positive. Any task for which this difficulty difference is not true does not satisfy the unmanageability criterion of the definition.

example, Kong, Huffman Hayes, Dekhtyar, and Holden show that humans are much better at validating tool-proposed trace links than they are at finding links in documents [10]. See the end of Section V for numerical empirical evidence, for a variety of hairy tasks, that manually finding a correct answer requires more time than rejecting as nonsense an incorrect answer.

Therefore, the evaluation of a tool for a hairy task should probably weight recall more than precision. However as demonstrated in Section VIII-C, a lot of the RE and SE literature about tools for hairy tasks weights precision the same as or more than recall to evaluate tools for hairy tasks.

Why is there such an emphasis on precision? Precision is important in the information retrieval (IR) area from which are borrowed many of the algorithms used to construct the tools for hairy tasks [11], [28]. In IR, users of a tool with low precision are turned off by having to reject false positives more often than they accept true positives. In some cases, only a few or even only one true positive is needed. Perhaps the force of habit drives people to evaluate the tools for hairy tasks with the same criteria that are used for IR tools. Also, "precision" sounds so much more important than "recall", as in "This output is precisely right!".

In fact, the most senior author of each of two of the works cited in VIII as having used the F-measure, which weights recall and precision equally, or as having emphasized precision more than recall, expressed surprise when I told him personally that his paper had done so. Each knew that recall was more critical than precision for his paper's task, and each claimed that his paper reflected or accounted for that understanding. However, after I showed each the relevant text in his paper, he agreed that a reasonable reader would interpret the text as I had. Apparently, some other author of his paper had just followed the evaluation convention inherited from IR in writing parts of the paper.

When I asked one author of each of two other papers why his paper had used the F-measure, he replied that doing so was conventional. When I asked him if he and his co-authors had considered the possibility that recall should be weighted more than precision, he said simply "No." One of the two added that he and his co-authors had enough else to worry about in conducting their tool evaluation.

In summary, a careful consideration of the requirements for tools for hairy RE and SE tasks makes it clear that measures that may be proper in IR may not apply when making tools for hairy RE and SE tasks. The requirements for each hairy task must be considered carefully, and if these requirements say that close to 100% recall is needed, then the main goal for its tools is achieving high recall.

IV. Other Sightings of the Same Ideas

There is another sighting of the concept of hairy task, albeit not with the same name. Menzies, Dekhtyar, Distefano, and Greenwald [32] discuss recall versus precision in the context of tools for detection of defects in CBS development. They describe industrial tool-use situations in which high recall is needed, and not only is low precision acceptable, but it may be *needed* in order to achieve high recall in an algorithmic tradeoff (See Section VII about this tradeoff.). Briefly, these situations are when

- the cost of missing a true positive is prohibitive,
- the true positives constitute only a small fraction of the input presented to the tool, and
- there is little or no cost to deciding that an answer from the tool is a false positive,

The first point captures the situation when the CBS has HD requirements, The second point captures both the unmanageability of the tool's hairy task and the difficulty of finding a true positive, and the third point captures that manually rejecting a false positive presented by the tool is cheap compared to manually finding a true positive. They observe that the second point characterizes many SE datasets. If this observation is correct, then it is clear that many SE tasks are unmanageable and are therefore hairy.

Not all instances of hairy tasks for which close to 100% recall is needed are in RE or SE. The annual Text REtrieval Conference (TREC) [33] has a Total Recall Track [34], whose statement of purpose lists three such tasks [35], [36] (enumeration not in the original).

> The primary purpose of the Total Recall Track is to evaluate, through controlled simulation, methods designed to achieve very high recall — as close as practicable to 100% — with a human assessor in the loop. Motivating applications include, among others, [1] electronic discovery in legal proceedings ..., [2] systematic review in evidence-based medicine ..., and [3] the creation of fully labeled test collections for information retrieval ("IR") evaluation ....

V. Weighted F-Measure

For situations in which recall and precision are not equally important, there is a weighted version of the F-measure,

$$F_\beta \;=\; (1+\beta^2) \times \frac{P \times R}{(\beta^2 \times P) + R} \quad , \tag{4}$$

called the F_β-measure.

Note that the simple F-measure is F_1. Note also that the formula for F_β ends up multiplying P by the dominating β^2 in both the numerator and the denominator. Thus, as β grows, very quickly F_β approaches R, and P becomes irrelevant in computing F_β. When β is as little as 5, and P is large enough relative to R, P is already essentially irrelevant.

In these formulae, β is the ratio by which it is desired to weight R more than P. How should β be determined? It should be calculated as the ratio of

- the time for a human to manually find a true positive in the original documents and
- the time for a human to reject a tool-presented false positive[4].

[4]on the assumption that the time required for a run of the tool is negligible or other work can be done while the tool is running on its own.

Some want to adjust this β according to the ratio of two other values,

- an estimate of the cost of the failure to find a true positive and
- an estimate of the cost of the accumulated nuisance of dealing with tool-presented false positives.

For any particular hairy task and a tool for it, a separate empirical study is necessary to arrive at good estimates for these values.

There is empirical evidence for any of a variety of hairy tasks that β is greater than 1, and in many cases, significantly so. For example, Section IX shows an estimate for β for the tracing task and one tracing tool [37] as 73.60. Section IX-B of the full paper [31] shows estimates for βs for the three hairy tasks [17] the section discusses as 10.00, 9.09, and 2.71. Tjong, in doing her evaluation of SREE, an ambiguity finder, found data that give a β of 8.7 [24].

Cleland-Huang *et al* calculate the returns on investment and costs vs. benefits of several tracing strategies ranging from maintaining full traces for immediate use at any time through tracing on the fly. To come to their conclusions, they estimated, probably based on their extensive experience with tracing, that manually finding a trace link between two documents takes on average 15 minutes [38]. Even though one of their tracing strategies involves use of a tool to generate traces on the fly, they give no estimate at all for the time to vet a tool-reported potential link, and estimate total costs of strategies without considering any costs associated with tool use. Therefore, they must regard that time as negligible. To be conservative, let us assume that the time to vet a single potential link is 1 minute. These two estimates yield an estimated β for tracing of 15.

Heindl and Biffl timed manual after-development (rather than during-development) identifications of a variety of trace links between requirements and code documents and obtained times ranging from 10 to 45 minutes, depending on the grainedness of the link [39]. They too give no estimate of the time to vet a tool-reported potential link. Again, to be conservative, let us assume that the time to vet a single potential link is 1 minute. These two estimates yield estimated βs for tracing ranging from 10 to 45.

VI. An Empirical Method to Evaluate Tools for NL RE Task

In order to evaluate a tool t for a NL RE task T to be performed on the documents D for the development of a CBS, two questions need to be answered.

1) One question is the obvious: "What is the relative importance of recall and precision, i.e., what is the correct value of β to use in F_β?'
2) The second is a deeper question, "What is the best method to perform T: entirely manually, with only t, or with some mixture thereof?" [40]

The answers to both of these questions can come only empirically, e.g., by adapting IR's cost-based evaluation measures [41] to the RE context.

To answer these questions, we need to determine:

1) *numerator of* β: the average time that an average human needs to manually find a correct answer in D, and
2) *denominator of* β: (1) the average time that an average human needs to manually vet any potential answer that t returns or (2) the average time that an average human needs to manually determine whether or not any potential answer in D is a correct answer [42].
3) *cost of a false negative*: the criticality of achieving 100% recall on T, by estimating the cost of a false negative, an undetected correct answer,
4) *cost of low precision*: the criticality of high precision, by estimating the tool-use deterrence created by each false positive reported by t,
5) *average tool recall*: the average recall that t achieves on T, and
6) *HAHR*: the average recall that humans achieve when they do T manually.

Many of these data can be obtained during a multi-person construction of a gold standard G of correct answers from manually performing T on a representative and substantial sampling of documents D from the construction of a representative CBS [43]. Generally, G is constructed by a group of people familiar with the CBS or its domain:

1) Each member of the group performs T on D independently to produce his or her own list of answers that he or she believes are correct.
2) The union of the group members' lists is formed.
3) The group meets to discuss the union list and its members' lists in order to arrive at a mutually-agreed-upon single list which is some variation of the union list.

The mutually-agreed-upon list is taken as G. Gold standard construction has its own problems [15] that have to be considered to ensure that evaluations based on it are valid.

During the construction of G, each group member keeps track of the total time he or she spent performing T on D, the number of correct answers he or she found, and the number of potential answers he or she examined. These data allow estimating (1) the numerator of β and (2) the denominator of β by the *second* method. The average human recall is the average of the fractions of G that were found by the members of the group.

Other data can be obtained during a test of a tool t applied to D. First, the recall and precision for t can be estimated by vetting the output of t with G. In addition, the person doing the vetting keeps track of the time he or she spends on the vetting. These times allow estimating the denominator of β by the *first* method. Note that the *second* method of estimating the denominator of β, used in the previous paragraph, gives an estimate for β that is independent of any tool and that can be obtained from only the gold-standard construction. Vetting of a tool-provided answer probably is faster than considering a potential answer during gold-standard construction, because the tool presents actual potential answers while considering a potential answer gold-standard construction requires looking at

the part of D from where the potential answer comes. Thus, the second method of estimating the denominator of β likely gives a lower bound for values obtained by the first method of estimating the denominator.

The cost of a false negative and the cost of low precision will have to be estimated by considering the context in which T is performed. Cleland-Huang *et al* and Heindl and Biffl suggest a number of risk-based strategies for estimating the cost of missing link on the CBS development in which traces are used to track down the impacts of requirements changes [38], [39].

VII. Summarization

For many a task for which there exists a variety of algorithms to implement it, the difference between any pair of algorithms amounts to a tradeoff between recall and precision. Generally, one can get higher recall at the expense of lower precision and vice versa. The extremes of this tradeoff are:

- the tool delivers the entire document for 100% recall and $C\%$ precision, where $C\%$ is the fraction of the answers in the document that are correct, and
- the tool delivers one correct answer for 100% precision and $\epsilon\%$ recall, where $\epsilon\%$ is the reciprocal of the number of correct answers in the document.

These extremes are useless, because in either case, the entire document has to be manually searched in order to find the correct answers.

However, one way out of this uselessness is offered by a new measure, *summarization*, which is the fraction of the original document that is eliminated in what the tool finds and delivers to its user[5]. If for a typical run of a tool on a document d, the size of the output o of the tool is $z\%$ of the size of d, then the summarization of the tool is $1-z\%$. In terms of the subspaces of the universe of a tool in Figure 1, summarization, S, can be defined:

$$\begin{aligned} S &= \frac{|\sim \mathsf{fnd}|}{|\sim \mathsf{fnd} \cup \mathsf{fnd}|} = \frac{|\sim \mathsf{fnd}|}{|\sim \mathsf{rel} \cup \mathsf{rel}|} \\ &= \frac{|\mathrm{TN}| + |\mathrm{FN}|}{|\mathrm{TN}| + |\mathrm{FN}| + |\mathrm{TP}| + |\mathrm{FP}|} \end{aligned} \tag{5}$$

A tool for a hairy task with 100% recall and 90% summarization is very helpful. In this case, the output of the tool contains all the sought answers. Because the tool's output is significantly smaller than its original input document, a manual search of the output is significantly easier than a manual search of the original document. Thus, the precision of the tool is effectively irrelevant; it could just as well be 0%. So, if an algorithm for a tool achieves high recall at the cost of low precision, but it summarizes a lot, then we should make use of it in building the tool.

Another, implemented, example of how summarization can be used in considering a tradeoff is offered by Montgomery and Damian [44]. They evaluate a machine-learning algorithm for the hairy task of determining from the natural language complaints of a program's customers the probability of those customers' escalating their complaints. The algorithm returns complaints called "CritSits" that it believes have a high probability of escalation. They were able to determine that the algorithm has 79.94% recall and 80.77% summarization. They observe, "Simply put, if a support analyst wanted to spend time identifying potential CritSits from PMRs, our model reduces the number of candidate PMRs by 80.77%, with the statistical guarantee that 79.94% of CritSits remain."

VIII. Related Work

There are lots of evaluations of tools for hairy RE and SE tasks. Some *are* basing their evaluations on mainly recall. A few have recognized the importance of recall, and recognize that F_1 is inappropriate, but are using only F_2, which does not weight recall according to its cost. However, even recently, some are basing their evaluations on mainly precision or F_1.

A. Basing Evaluation on Recall

Many developers of tools for hairy tasks *have* recognized that recall is more important for their task than precision and have based their evaluations on recall [9], [21], [23], [45]–[49]. However, one such team of tool developers says that its goal was 100% recall, but it accepted much lower recall as satisfactory, arguing that humans do about the same [21].

B. Basing Evaluation on F_2

Some developers of tools for hairy tasks *have* recognized that recall is more important for their task than precision and have based their evaluations on F_2 [23], [45], [49], [50]. The results of each cited work, which used F_2, or a variation, to come to a favorable conclusion about its tool can be strengthened by the use of F_β, with β being 5, 10, or even larger. See Section IX-D of the full paper for a detailed reevaluation of the results of Arora *et al* [49].

C. Basing Evaluation on Precision or F_1

However, even as late as in 2017, there are still some developers of hairy task tools who appear to regard precision as at least as important as recall, at least part of the time, and that evaluate their tools with precision or with F_1. See the full paper [31] for a list of 16 works doing so.

IX. Example Reevaluations

Section IX of the full paper [31] examines several recent publications in which this author believes that F_1 was used when recall should have been weighted more than precision, and thus, that F_β should have been used with a β greater

[5]Summarization for a tool is meaningful only if the output of the tool is in the same language as its input. For example, summarization is meaningful for an ambiguity finder which returns each sentence in the input natural language sentence that the tool believes has an ambiguity. In this case, the language of the input and of the output is the set of natural language sentences. Summarization is not meaningful for a trace link finder that given multiple documents of natural language sentences returns what it believes to be links between related sentences in the documents. In this case, the input language is the set of natural language sentences and the output language is the set of links between sentences. The point of summarization is that after the tool is finished with its attempt at a hairy task, the human user can carry out the same task on the output of the tool, which is in the same language as the input to the tool.

The highlighted calculation is wrong. It's 4 times larger than it should be, 18.40, and the whole boxed text should be replaced by what is on the additional, LAST page.

than 1. For each such publication, its results are reexamined using an F_β, with the value of β determined empirically in some cases. Most of these reevaluations end up strengthening or improving the results of the examined papers.

To meet the space limitations for this paper, only the first of these reevaluations, that of tracing tools by Merten *et al*, is included herein. The omitted material includes detailed reevaluations of various tools for categorizing app reviews by Maalej, Kurtanović, Nabil, and Stanik [17] in Section IX-B and of tools for extracting feature-relevant information from users' manuals by Quirchmayr, Paech, Kohl, and Karey [51] in Section IX-C.

TABLE I
COMPUTED F_β VALUES

Data Source	P	R	F_1	F_2	F_{73}
Related Work Low P	0.1	0.9	0.18	0.35	0.898
Related Work High P	0.2	0.9	0.33	0.53	0.899
Merten *et al*	0.02	1.0	0.039	0.093	0.990

Merten, Krämer, Mager, Schell, Bürsner, and Paech extensively evaluated the performance of five different IR algorithms in the task of extracting trace links for a CBS from the data in the issue tacking system (ITS) for the CBS's development [37]. They calculated the recall and the precision of each algorithm by comparing its output for any ITS with a gold standard set of links extracted manually from the ITS. In order that both recall and precision be taken into account in the evaluation, they calculated F_1 and F_2, the latter because it "emphasizes recall". However, as is shown below, F_2 does not emphasize recall enough.

After Merten *et al* state that their goal is to "maximize precision, recall, F_1 and F_2 measure", their "Results are presented as F_2 and F_1 measure in general." They summarize their results:

> In terms of algorithms, to our surprise, no variant of BM25 competed for the best results. The best F_2 measures of all BM25 variants varied from 0.09 to 0.19 over all projects, independently of standard preprocessing. When maximizing R to 1, P does not cross a 2% barrier for any algorithm. Even for $R \geq 0.9$, P is still < 0.05. All in all, the results are not good according to [standards set by Hayes, Dekhtyar, and Sundaram [9]] independently of standard preprocessing, and they cannot compete with related work on structured RAs [requirements artifacts].

This summary is puzzling because, earlier they had said,

> However, maximising [sic] recall is often desirable in practice, because it is simpler to remove wrong links manually than to find correct links manually.

If so, Merten *et al* should be ecstatic that at least one of their algorithms is able to achieve an R of 1.0, *even* at the expense of a P of 0.02! This recall is better than the best that they gave in their "Related Work" section, that reported by Gotel et al [12]:

> "[some] methods retrieved almost all of the true links (in the 90% range for recall) and yet also retrieved many false positives (with precision in the low 10–20% range, with occasional exceptions)."

That is, the best R is around 0.9 with P between 0.1 and 0.2.

Table I shows in Columns 4 and 5, the F_1 and F_2 values computed from the three pairs of P and R values that they compared. Certainly, each of F_1 and F_2 for the Related-Works $R = 0.9$ and $P = 0.1$ or $P = 0.2$ is about an order of magnitude bigger than that for the Merten-*et-al* $R = 1.0$ and $P = 0.02$.

Merten *et al* give some data from which it is possible to estimate β. Merten *et al* say that they manually created gold standard trace matrices (GSTMs). From the paragraph titled "Gold Standard Trace Matrices" in their Section 5.1, it is revealed that for each project, three of the authors together made 4950 manual comparisons over 2.5 person 8-hour (confirmed by e-mail with Merten) business days. Thus, the total time spent on these comparisons per project is $2.5 \times 8 = 20$ person hours $= 1200$ person minutes. The same paragraph reveals that in these 1200 person minutes per project, 4950 comparisons were done. Thus, per project, the average comparison required $\frac{1200}{4950}$ person minutes per comparison, or 0.2424 person minutes per comparison. So the three authors were deciding whether a potential link was or was not a true link very quickly, in about 14.54 seconds.

Table 2 of their paper shows in the "GSTM generic" row, the number of links found for the four projects. They are 102, 18, 55, and 94, for a total of 269 links. The total time spent in comparisons over the four projects is $4 \times 1200 = 4800$ person minutes. Thus each link required $\frac{4800}{269} = 17.84$ person minutes to find.

These data say that β should be $\frac{17.84}{0.2424} = 73.60$, much larger than 10. Let's round 73.60 downward to 73. Column 6 of Table I shows F_{73} values for the same pairs of Ps and Rs With F_{73}, the R and P of Merten *et al* clearly outperform the R and P of either of the Related Works. When $\beta = 73$, an R of 1.0 is *truly* better than an R of 0.9, regardless of the P value, if close to 100% recall is essential.

This empirically based estimate for β of 73.60 is much larger than the 10 that this author had been using based on his feeling that for a hairy task, manually finding a true positive takes an order of magnitude more time than manually rejecting a false positive. That this estimate is so high gives this author confidence that β will be higher than expected for many hairy tasks and that conclusions that are claimed on the basis of even F_5 (Recall the discussion about Formula 4.) in the full paper are valid.

X. HUMAN FACTORS IN TOOL USE

More and more builders of tools for hairy tasks are evaluating their tools not in isolation, but by comparing humans

working with the tool on documents with humans working manually on the same documents [8], [30], [52]–[55].

The evaluation of a tool needs to consider the willingness of humans to use the tool. Huang *et al* do warn that [45]

> Unfortunately, prior studies have demonstrated that users lose confidence in a traceability tool that returns imprecise results. Furthermore human error is introduced when human analysts are asked to evaluate a long list of candidate links ....

while citing [56].

This clear drawback to tools with any appreciable imprecision will have to be addressed. The cost of low precision in Section VI is an attempt to measure this drawback. It may be necessary to repeatedly remind the user of any high-recall, low-precision tool for a hairy task being applied to a CBS with HD requirements of (1) the relative costs for this task of finding a true positive manually and of rejecting a tool-provided false positive, (2) the tool's recall compared with the task's HAHR, (3) the tool's summarization, and (4) the costs of the manual alternatives.

In any case, if an important hairy task is to be performed on NL documents for a life-critical CBS; there is a tool for the task whose empirically determined β is, say, 50; the tool's recall is significantly higher than the task's HAHR, as measured by domain experts; but the tool's precision is low, then the developers' failure to use the tool, even in the face of loud complaints about its low precision, is unethical and leaves the developers liable for malpractice for failure to apply known best practices when human lives are at stake.

XI. Conclusions

Most RE and SE tasks involving NL documents are hairy tasks, and therefore, they beg for tool support. Some of these tools must achieve the task's HAHR, close to 100% recall, especially when the task is being conducted on the documents of a CBS with HD requirements.

Without carefully considering the requirements for a tool for a hairy RE or SE task, we, in the RE and SE fields, have evaluated many of these tools with the same recall, precision, and F-measure that are used in the NLP and IR fields. In those fields, for many a task, precision is more important. For a different many a task, precision is as important as recall; so F_1 is the correct composite of recall and precision. However, there are hairy RE and SE tasks that might be performed in the context of a HD CBS, and for many of them, recall is critical. Moreover, for the typical of these tasks, the ratio of the time to manually find a true positive over the time to manually reject a false positive is probably at least 10. Even better is to empirically determine this ratio for the task and to use this value as β in F_β.

Our habit of using traditional measures by default must *stop*. We, in effect, must do RE for each hairy task, to understand which measures are appropriate to evaluate any tool for the task. We must then do the empirical studies that are needed to obtain all values, including the HAHR for the task, β for the task or for the tool, the various costs, the times for steps of the gold-standard construction process, etc., that are used in these measures. Finally, we must use these measures to evaluate the tool. If, in the end, F_{10} is determined to be appropriate, then using F_2 is only a step in the right direction.

Acknowledgments

The author benefited from discussions with T. Breaux, L. Briand, J. Cleland-Huang, A. Ferrari, W. Maalej, A. Massey, T. Merten, and J. Mylopoulos, and his work was supported in part by grant NSERC-RGPIN227055-00 (Canada).

References

[1] K. Ryan, "The role of natural language in requirements engineering," in *Proceedings of the IEEE International Symposium on Requirements Engineering*, 1993, pp. 240–242.

[2] L. Northrop, B. Pollak, P. Feiler, R. P. Gabriel, J. Goodenough, R. Linger, T. Longstaff, R. Kazman, M. Klein, D. Schmidt, K. Sullivan, and K. Wallnau, *Ultra-Large-Scale Systems, The Software Challenge of the Future*. Pittsburg, PA, USA: Software Engineering Institute at Carnegie Mellon University, 2006. [Online]. Available: http://www.sei.cmu.edu/library/assets/ULS_Book20062.pdf

[3] L. Goldin and D. M. Berry, "AbstFinder: A prototype abstraction finder for natural language text for use in requirements elicitation," *Automated Software Engineering*, vol. 4, pp. 375–412, 1997.

[4] R. Gacitua and P. Sawyer, "Ensemble methods for ontology learning - an empirical experiment to evaluate combinations of concept acquisition techniques," in *ICIS'2008*, 2008, pp. 328–333.

[5] R. Gacitua, P. Sawyer, and V. Gervasi, "On the effectiveness of abstraction identification in requirements engineering," in *Proceedings of the 18th IEEE International Requirements Engineering Conference (RE)*, 2010, pp. 5–14.

[6] A. Dwarakanath, R. R. Ramnani, and S. Sengupta, "Automatic extraction of glossary terms from natural language requirements," in *Proceedings of the IEEE International Requirements Engineering Conference (RE)*, 2013, pp. 314–319.

[7] M. Rahimi, M. Mirakhorli, and J. Cleland-Huang, "Automated extraction and visualization of quality concerns from requirements specifications," in *Proceedings of the IEEE International Requirements Engineering Conference (RE)*, 2014, pp. 253–262.

[8] N. Zeni, N. Kiyavitskaya, L. Mich, J. R. Cordy, and J. Mylopoulos, "Gaiust: Supporting the extraction of rights and obligations for regulatory compliance," *Requirements Engineering Journal*, vol. 20, no. 1, pp. 1–22, 2015.

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

[10] W. Kong, J. H. Hayes, A. Dekhtyar, and J. Holden, "How do we trace requirements: An initial study of analyst behavior in trace validation tasks," in *Proceedings of the International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, 2011, pp. 32–39.

[11] A. D. Lucia, A. Marcus, R. Oliveto, and D. Poshyvanyk, "Information retrieval methods for automated traceability recovery," in *Software and Systems Traceability*, J. Cleland-Huang, O. Gotel, and A. Zisman, Eds. Springer, 2012, pp. 71–98.

[12] O. Gotel, J. Cleland-Huang, J. H. Hayes, A. Zisman, A. Egyed, P. Grünbacher, and G. Antoniol, "The quest for ubiquity: A roadmap for software and systems traceability research," in *2012 20th IEEE International Requirements Engineering Conference (RE)*, 2012, pp. 71–80.

[13] V. Gervasi and D. Zowghi, "Supporting traceability through affinity mining," in *Proceedings of the IEEE International Requirements Engineering Conference (RE)*, 2014, pp. 143–152.

[14] K. Wnuk, M. Höst, and B. Regnell, "Replication of an experiment on linguistic tool support for consolidation of requirements from multiple sources," *Empirical Softw. Engg.*, vol. 17, no. 3, pp. 305–344, 2012.

[15] G. Cavalcanti, P. Borba, and P. Accioly, "Should we replace our merge tools?" in *Proceedings of the 39th International Conference on Software Engineering Companion (ICSE-C)*, 2017, pp. 325–327.

[16] A. Casamayor, D. Godoy, and M. Campo, "Functional grouping of natural language requirements for assistance in architectural software design," *Knowledge-Based Systs.*, vol. 30, pp. 78–86, 2012.

[17] W. Maalej, Z. Kurtanović, H. Nabil, and C. Stanik, "On the automatic classification of app reviews," *Requirements Engineering Journal*, vol. 21, no. 3, pp. 311–331, 2016.

[18] N. Jha and A. Mahmoud, "Mining user requirements from application store reviews using frame semantics," in *Proceedings of the 23rd International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2017)*, 2017, pp. 273–287.

[19] M. Robeer, G. Lucassen, J. M. E. van der Werf, F. Dalpiaz, and S. Brinkkemper, "Automated extraction of conceptual models from user stories via NLP," in *Int. Reqs. Engg. Conf. (RE)*, 2016, pp. 196–205.

[20] F. Chantree, B. Nuseibeh, A. de Roeck, and A. Willis, "Identifying nocuous ambiguities in natural language requirements," in *Proceedings of the IEEE International Requirements Engineering Conference (RE)*, 2006, pp. 56–65.

[21] B. Gleich, O. Creighton, and L. Kof, "Ambiguity detection: Towards a tool explaining ambiguity sources," in *Proceedings of the International Working Conference on Requirements Engineering: Foundation of Software Quality (REFSQ)*, 2010, pp. 218–232.

[22] A. P. Nikora, J. H. Hayes, and E. A. Holbrook, "Experiments in automated identification of ambiguous natural-language requirements," in *Proceedings of the International Symposium on Software Reliability Engineering*, 2010, pp. 229–238.

[23] H. Yang, A. N. D. Roeck, V. Gervasi, A. Willis, and B. Nuseibeh, "Analysing anaphoric ambiguity in natural language requirements," *Requirements Engineering Journal*, vol. 16, no. 3, pp. 163–189, 2011.

[24] S. F. Tjong and D. M. Berry, "The design of SREE — A prototype potential ambiguity finder for requirements specifications and lessons learned," in *Proceedings of the International Working Conference on Requirements Engineering: Foundation of Software Quality (REFSQ)*, 2013, pp. 80–95.

[25] F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami, "An automatic quality evaluation for natural language requirements," in *Reqs. Engg.: Foundation Softw. Quality (REFSQ)*, 2001, pp. 1–18.

[26] W. M. Wilson, L. H. Rosenberg, and L. E. Hyatt, "Automated analysis of requirement specifications," in *Proc. 19th International Conference on Software Engineering (ICSE)*, 1997, pp. 161–171.

[27] A. Bucchiarone, S. Gnesi, and P. Pierini, "Quality analysis of NL requirements: An industrial case study," in *Proc. 13th IEEE International Requirements Engineering Conference (RE)*, 2005, pp. 390–394.

[28] D. M. Berry, R. Gacitua, P. Sawyer, and S. F. Tjong, "The case for dumb requirements engineering tools," in *Proceedings of the International Working Conference on Requirements Engineering: Foundation of Software Quality (REFSQ)*, 2012, pp. 211–217.

[29] P. Mäder and A. Egyed, "Do developers benefit from requirements traceability when evolving and maintaining a software system?" *Empirical Software Engineering*, vol. 20, no. 2, pp. 413–441, 2015.

[30] T. Saracevic, "Evaluation of evaluation in information retrieval," in *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 1995, pp. 138–146.

[31] D. M. Berry, "Evaluation of tools for hairy requirements engineering and software engineering tasks," School of Computer Science, University of Waterloo, Tech. Rep., 2017. [Online]. Available: https://cs.uwaterloo.ca/~dberry/FTP_SITE/tech.reports/EvalPaper.pdf

[32] T. Menzies, A. Dekhtyar, J. Distefano, and J. Greenwald, "Problems with precision: A response to "comments on 'data mining static code attributes to learn defect predictors'"," *IEEE Transactions on Software Enginerring*, vol. 33, no. 9, pp. 637–640, 2007.

[33] TREC Conferences, "Text REtrieval Conference (TREC)," viewed on 8 February 2017, http://trec.nist.gov.

[34] ——, "TREC 2015 total recall track," 2015, http://plg.uwaterloo.ca/~gvcormac/total-recall/guidelines.html.

[35] A. Roegiest, G. V. Cormack, M. R. Grossman, and C. L. Clarke, "TREC 2015 total recall track overview," 2016, http://trec.nist.gov/pubs/trec24/trec2015.html.

[36] M. R. Grossman, G. V. Cormack, and A. Roegiest, "TREC 2016 total recall track overview," 2016, http://trec.nist.gov/pubs/trec25/trec2016.html.

[37] T. Merten, D. Krämer, B. Mager, P. Schell, S. Bürsner, and B. Paech, "Do information retrieval algorithms for automated traceability perform effectively on issue tracking system data?" in *Proceedings of the 22nd International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ)*, 2016, pp. 45–62.

[38] J. Cleland-Huang, G. Zemont, and W. Lukasik, "A heterogeneous solution for improving the return on investment of requirements traceability," in *Proceedings of the 12th IEEE International Requirements Engineering Conference (RE)*, 2004, pp. 230–239.

[39] M. Heindl and S. Biffl, "A case study on value-based requirements tracing," in *Proceedings of the 10th European Software Engineering Conference Held Jointly with 13th ACM SIGSOFT International Symposium on Foundation for Software Engineering ESEC/FSE*, 2005, pp. 60–69.

[40] D. M. Berry, A. Ferrari, and S. Gnesi, "Assessing tools for defect detection in natural language requirements: Recall vs precision," School of Computer Science, University of Waterloo, Tech. Rep., 2017. [Online]. Available: https://cs.uwaterloo.ca/~dberry/FTP_SITE/tech.reports/BFGpaper.pdf

[41] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*. Burlington, MA, USA: Morgan Kaufmann, 2016.

[42] W. Maalej, "Private communication," March 2017.

[43] C. D. Manning, P. Raghavan, and H. Schütze, "Chapter 8: Evaluation in information retrieval," in *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press, 2008. [Online]. Available: https://nlp.stanford.edu/IR-book/pdf/08eval.pdf

[44] L. Montgomery and D. Damian, "What do support analysts know about their customers? on the study and prediction of support ticket escalations in large software organizations," in *Proceedings of the 25th IEEE International Requirements Engineering Conference (RE)*, 2017, p. to appear.

[45] J. Cleland-Huang, A. Czauderna, M. Gibiec, and J. Emenecker, "A machine learning approach for tracing regulatory codes to product specific requirements," in *Proceedings of the International Conference on Software Engineering ICSE)*, 2010, pp. 155–164.

[46] J. Cleland-Huang, O. Gotel, and A. Zisman, Eds., *Software and Systems Traceability*. London, UK: Springer, 2012.

[47] C. Ingram and S. Riddle, "Cost-benefits of traceability," in *Software and Systems Traceability*, J. Cleland-Huang, O. Gotel, and A. Zisman, Eds. Springer, 2012, pp. 23–42.

[48] A. Mahmoud and N. Niu, "Supporting requirements to code traceability through refactoring," *Requirements Engineering Journal*, vol. 19, no. 3, pp. 309–329, 2014.

[49] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer, "Automated checking of conformance to requirements templates using natural language processing," *IEEE Transactions on Software Engineering*, vol. 41, no. 10, pp. 944–968, 2015.

[50] A. Delater and B. Paech, "Tracing requirements and source code during software development: An empirical study," in *Proceedings of the International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2013, pp. 25–34.

[51] T. Quirchmayr, B. Paech, R. Kohl, and H. Karey, "Semi-automatic software feature-relevant information extraction from natural language user manuals," in *Proceedings of the International Working Conference on Requirements Engineering: Foundation of Software Quality (REFSQ)*, 2017, p. to appear.

[52] D. Cuddeback, A. Dekhtyar, and J. H. Hayes, "Automated requirements traceability: The study of human analysts," in *Proceedings of the IEEE International Requirements Engineering Conference (RE)*, 2010, pp. 231–240.

[53] P. R. Anish and S. Ghaisas, "Product knowledge configurator for requirements gap analysis and customizations," in *Proceedings of the IEEE International Requirements Engineering Conference (RE)*, 2014, pp. 437–443.

[54] A. Ferrari, F. Dell'Orletta, G. O. Spagnolo, and S. Gnesi, "Measuring and improving the completeness of natural language requirements," in *Proceedings of the International Working Conference on Requirements Engineering: Foundation of Software Quality (REFSQ)*, 2014, pp. 23–38.

[55] E. Knauss and D. Ott, "(Semi-) automatic categorization of natural language requirements," in *Proceedings of the International Working Conference on Requirements Engineering: Foundation of Software Quality (REFSQ)*, 2014, pp. 39–54.

[56] J. H. Hayes, A. Dekhtyar, S. K. Sundaram, and S. Howard, "Helping analysts trace requirements: An objective look," in *Proceedings of the 12th IEEE International Requirements Engineering Conference (RE)*, 2004, pp. 249–259.

The boxed text is the correction from an enhancement of the tech report and workshop paper. \beta_{T} is what is called \beta estimated by the second method, for the task, independent of any tool, and \beta_{Tt} is what is called \beta calculated by the first method, specific for each tool.

with a gold standard set of links extracted manually from the ITS. In order that both recall and precision be taken into account in the evaluation, they calculated F_1 and F_2, the latter because it "emphasizes recall". However, as is shown below, F_2 does not emphasize recall enough.

After Merten *et al* state that their goal is to "maximize precision, recall, F_1 and F_2 measure", their "Results are presented as F_2 and F_1 measure in general." They summarize their results:

> In terms of algorithms, to our surprise, no variant of BM25 competed for the best results. The best F_2 measures of all BM25 variants varied from 0.09 to 0.19 over all projects, independently of standard preprocessing. When maximizing R to 1, P does not cross a 2% barrier for any algorithm. Even for $R \geq 0.9$, P is still < 0.05. All in all, the results are not good according to [standards set by Hayes, Dekhtyar, and Sundaram [38]] independently of standard preprocessing, and they cannot compete with related work on structured RAs [requirements artifacts].

This summary is puzzling because, earlier they had said,

> However, maximising [sic] recall is often desirable in practice, because it is simpler to remove wrong links manually than to find correct links manually.

If so, Merten *et al* should be ecstatic that at least one of their algorithms is able to achieve an R of 1.0, *even* at the expense of a P of 0.02! This recall is better than the best that they gave in their "Related Work" section, that reported by Gotel et al [30]:

> "[some] methods retrieved almost all of the true links (in the 90% range for recall) and yet also retrieved many false positives (with precision in the low 10–20% range, with occasional exceptions)."

That is, the best R is around 0.9 with P between 0.1 and 0.2.

Table 6 shows in Columns 4 and 5, the F_1 and F_2 values computed from the three pairs of P and R values that they compared. Certainly, each of F_1 and F_2 for the Related-Works $R = 0.9$ and $P = 0.1$ or $P = 0.2$ is about an order of magnitude bigger than that for the Merten-*et-al* $R = 1.0$ and $P = 0.02$.

Merten *et al* give some data from which it is possible to estimate β. Merten *et al* say that they manually created gold standard trace matrices (GSTMs). From the paragraph titled "Gold Standard Trace Matrices" in their Section 5.1, it is revealed that for each project, three of the authors together made 4950 manual comparisons. Table 2 of their paper shows in the "GSTM generic" row, the number of links found for the four projects. They are 102, 18, 55, and 94, for a total of 269 links. Thus, for this context, $\lambda = \frac{269}{4950} = 0.054$, and $\beta_T = \frac{4950}{269} = 18.40$, nearly twice as large as 10.

Let's round 18.40 downward to 18. Columns 6 and 7 of Table 6 show F_{18} and F_{21} values for the same pairs of Ps and Rs. With F_{18}, the R and P of Merten *et al* just underperform the R and P of both of the Related Works. With F_{21}, the R and P of Merten *et al* just outperform the R and P of either of the Related Works. The F_{18} is with β_T, the task β. If vetting a tool t's candidate link is only 14.13% faster than manually deciding a candidate link in situ in the documents, then β_{Tt} is 21. When $\beta = 21$, an R of 1.0 is *truly* better than an R of 0.9, regardless of the P value, if close to 100% recall is essential.

There are reasons to believe that $\frac{\beta_T}{\beta_{Tt}}$ is larger than 1 for the tracing task.

Note that because Merten *et al* describe a tool for the tracing taks, they escape the recall–precision tradeoff conundrum, as described in Section 6.

The Merten *et al* context seems to be one like that described by Hayes *et al*, when they observed that "In our prior work [citing [38]], we observed that even a high-recall, low-precision candidate TM [(trace matrix)] already generates savings as compared to the analyst's need to examine every pair of low-level/high-level elements when measured in terms of selectivity." [36].

This empirically calculated value for β_T of 18.40 is nearly twice the 10 that I had been using based on gut feelings. Moreover, this 18.40 is a β_T for the tracing *task*, independent of any tool t's β_{Tt}. If indeed vetting any t's output is faster than deciding on a candidate link *in situ* in the document, then any t's β_{Tt} will be larger than 18.40. Interestingly, Hayes *et al* simulation studies, the highest value of this paper's β used is 20; presumably, Hayes *et al* selected 20 based on their extensive experience with developing and evaluating tracing tools. That an empirical value for β is about double my gut-feeling estimate gives ne confidence that β will be higher than expected for many hairy tasks and that conclusions that are claimed on the basis of even F_5 in this paper are valid.