1

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 recall[4] (HAR)" for the task, which we hope is close to 100%. If a tool can be demonstrated to achieve better recall than the HAR 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 at least the HAR 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 [29,75,87]. Empirical studies will be needed to estimate the HAR and other key values that inform the evaluations.

[4]This used to be called the 'humanly achievable high recall (HAHR)", expressing the hope that it is close to 100%. However, actual values have proved to be quite low, sometimes as low as 32.95%.

In these formulae, β is the ratio by which it is desired to weight R more than P [38]. Call the β for a tool t for a task T "β_{Tt}". β_{Tt} should be calculated as the ratio of

numerator: the average time for a human, performing T manually, to find a true positive (i.e., correct) answer among all the potential answers in the original documents and

denominator: the average time for a human to determine whether or not an answer presented by t is a true positive answer[8].

The numerator can be seen as the human time cost of each item of recall, and the denominator can be seen as the human time cost of each item of precision.

Sometimes, one needs to estimate β for T *before* any tool has been built, e.g., to see if building a tool is worth the effort or to be able to make rational tradeoffs in building any tool. Call this task-dependent, tool-independent estimate "β_T". It uses the same numerator as β_{Tt} but a different denominator:

numerator: the average time for a human, performing T manually, to find a true positive answer among all the potential answers in the original documents and

denominator: the average time for a human, performing T manually, to decide whether or not any potential answer in the original document is a true positive answer.

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

The difference between the denominator and the numerator for β_T is that to find a true positive, one will have to decide about some a priori unknown number of potential answers to find one true positive answer, a number dependent on the incidence of true positive answers among the potential answers in the document. Let λ be the fraction of the potential answers in the document that are true positive answers. Then, β_T is $\frac{1}{\lambda}$ [50]. The less frequent the true positives are in a document, the hairier the task of finding them is.

In general, the denominator of a task's β_T is expected to be larger than the denominator of β_{Tt} for any well-designed t for T. A well-designed t will show for each potential true positive answer it offers, the snippets of the original document that it used to decide that the offered answer is potentially a true positive. These snippets should make deciding about an answer offered by t faster than deciding about the same answer while it is embedded in the original document. Thus, T's β_T should be a lower bound for the β_{Tt}s for all well-designed ts for T. In

the rest of this paper, "β" is a generic name covering both "β_T" and "β_{Tt}".

Some want to adjust 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-found false positives.

For any particular hairy task, a tool for it, and a context in which the task must be done, 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 8.4 shows a variety of estimates of β_T for the tracing task as 23.17, 22.70, 143.21, 23.65, 27.91, 57.05, and 18.40. Section 9.4 shows estimates for β_Ts for the three hairy tasks [51] 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_{Tt} of 8.7 [78].

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 the tracing task, T, that

- during the writing of the software being traced, creating a link takes on average 15 minutes and keeping any created link takes on average 5 minutes over five years of development, and
- when tracing on the fly is needed, e.g., during update of the software, finding a link manually takes on average 90 minutes [17].

Even though one of their tracing strategies involves use of a tool, t, to generate traces on the fly, they give no estimate at all for the time to vet a tool-found candidate link, and estimate total costs of strategies without considering any costs associated with tool use. Therefore, they must regard that time as negligible. If the vetting time is truly negligible, it must be in the seconds. Let us assume a conservative vetting time of 1 minute. These two times yield an estimate of $\beta_{Tt} = 90$ for the tracing tools Cleland-Huang *et al* were thinking of in their model.

6.2 Selectivity

For the tracing task, there is a phenomenon similar in effect to summarization. Suppose that the documents D to be traced consists of M items that can be the tail of a link and N items that can be the head of a link. Then, there are potentially $M \times N$ links, only a fraction of which are correct, true positive links. If a tool returns for vetting by the human user L candidate links, then the tool is said to have

$$\textit{selectivity} = \frac{L}{M \times N} \quad . \tag{7}$$

As Hayes, Dekhtyar, and Sundaram put it [38],

> " In general, when performing a requirements tracing task manually, an analyst has to vet M × N candidate links, i.e., perform an exhaustive search. Selectivity measures the improvement of an IR algorithm over this number: ... The lower the value of selectivity, the fewer links that a human analyst needs to examine."

Thus, *selectivity* is S, summarization, adapted to the tracing task[10]. If a tool for the tracing task has 100% recall and any selectivity strictly less than 100%, using the tool will offer *some* savings over doing the task manually, even if the precision is 0%. As is shown in Section 9.2, Sundaram, Hayes, and Dekhtyar found for various tracing tool algorithms, selectivity values in the range of 41.9% through 71.5% [76]. Therefore, the savings will be real.

[10] It is unfortunate that the senses of the summarization and selectivity measures are opposed to each other. A high summarization, near 100%, is good, and a low one, near 0%, is bad, while a low slectivity, near 0%, is good, and a high one, near 100%, is bad. Therefore, for clarity, the terms "good" and "bad" are used instead of "high" and "low" when talking about either.

There are other factors that indicate even greater savings by using a tool. While the output of a tracing tool is not in the same language as the input, the output, namely a list of candidate links, is physically much smaller than the input and is entirely focused on providing information to allow rapid vetting of the candidate links. A candidate link will show the snippets of the documents that are linked by the link. It may also show the data that led the tool declare the link to be a candidate.

As Barbara Paech observed in private communication [64],

> For me the value of the tool would be the organization. It takes notes of everything I have done. I cannot mix up things and so on.
>
> So I think the value is not so much per decision, but there is saving in the overall time.
>
> Furthermore I can imagine that the tool has other support. It could e.g. highlight for IR-created links the terms which are similar in the two artifacts. That would make the decision much easier."

It should take less time to vet a candidate link in a tracing tool's output than it does to manually find the same candidate link in the input and then to vet it, if for no reason other than the latter time includes the former. So, if one knows that the output of the tracing tool has recall greater than the tracing task's HAR, i.e., the tool beats the average human in finding true positive links, it really does not matter if the output has low precision, because in any case, the tool user will spend less total time to vet the tool's output than he or she would to do the tracing manually.

These observations suggest that in developing a tracing tool, it is probably best to trade away precision towards achieving recall as close as possible to 100% with decent selectivity of no higher than 70%. While users may balk at high imprecision [15], perhaps their being shown the evaluation data will convince them that the alternative of doing the task manually is *much much* worse!

7 An Empirical Method to Evaluate Tools for NL RE Task

In order to evaluate a tool for a NL RE task to be performed on some documents, 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?" [7]

The answers to both of these questions can come only empirically, e.g., by adapting IR's cost-based evaluation measures [83] to the hairy task context.

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, we need to determine as many of the following values as are possible and relevant:

1. *numerator of both* βs: the average time, τ_{find}, that an average human needs to manually find a correct answer in D,

2. *denominator of* β_{Tt}: the average time, τ_{vet}, that an average human needs to manually vet any potential answer that t returns,

3. *denominator of* β_T: the average time, τ_{det}, that an average human needs to manually determine whether or not any potential answer in D is a correct answer,

4. λ_D: the frequency of the true positive answers among all potential answers in D,

5. S or *selectivity*: the summarization of D achieved by t or, in the case T is tracing, the selectivity achieved by t on D,

6. *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,

7. *cost of low precision*: the criticality of high precision, by estimating the tool-use deterrence created by each false positive reported by t,

8. *average tool recall*: the average recall that t achieves on T, and

9. *HAR*: the average recall that humans achieve when they do T manually.

Note that as a consistency check or as a means to calculate a missing value, it should be that, or be assumed that

$$\frac{\tau_{find}}{\tau_{det}} = \frac{1}{\lambda_D} \quad . \tag{8}$$

For any of these value in which there are multiple data contributing to it, the average, minimum, maximum, and standard deviation of these data should be reported to allow fuller understanding of the estimated value.

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 [55]. 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 [13] that have to be considered to ensure that evaluations based on it are valid. Also, there are issues in the estimation of a HAR that should be investigated empirically. How is the value of a HAR affected by the experience of, the domain familiarity of, and the number of analysts building a gold standard?

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_T.

The average human recall is the average of the fractions of G that were found by the members of the group. This value can be taken as the HAR for T.

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. This estimated recall for t can be compared with the HAR for T. Second, S or *selectivity* of the output of t with respect to D can be computed.

In addition, the user of t doing the vetting keeps track of the time he or she spends on the vetting. These times allow estimating the denominator of β_{Tt}. If for t for T, $\tau_{vet} > \tau_{det}$, then using t to do T may be an impediment, and it might be better to do T manually.

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 [17,40]. Huang *et al*, Hayes *et al*, and Winkler and Vogelsang have found empirical evidence of the effect of a tool's low precision on the motivation, attentitiveness, and performance of the users of the tool [15, 39, 82].

MKMSBP give some data from which it is possible to compute β_T. They 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 MKMSBP just underperform the R and P of both of the Related Works. With F_{21}, the R and P of MKMSBP 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. See the last half of Section 6.2.

The MKMSBP R and P, being 100% and 2%, respectively, raises the spectre that the high recall is achieved by the extreme tradeoff of delivering the entire document, as is described in the beginning of Section 6. However, because MKMSBP's tool is for tracing, the tracing task, the output is in a language different from that of the input, this tradeoff is not possible. So, selectivity becomes the deciding measure. Even if selectivity is close to 100%, i.e., bad, vetting a link in a tool's output is usually faster than deciding the correctness of a potential link in a manual search. The MKMSBP context seems to be like that described by Hayes *et al*, when they observed that [36],

> 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.