# Who's Afraid of the Big Bad Imprecision? or How I learned to *love* trading lower precision for higher recall in tools for the hairiest tasks

Daniel M. Berry[0000−0002−6817−9081]

Cheriton School of Computer Science
University of Waterloo
Waterloo, ON N2L 3G1, Canada
dberry@uwaterloo.ca

18 June 2026

**Abstract.** *[Context and Motivation]* One is developing and evaluating a tool for performing a hairy natural-language-processing requirements-engineering task on the artifacts for the development of a critical computer-based system. So, doing the task cannot be avoided. A key characteristic of a hairy task is that the frequency of true positives in a tool's input is extremely low. Therefore, high recall is significantly more important to achieve than high precision. *[Problem]* In a tool, normally recall and precision trade off. So, achieving the goal of high recall often means accepting low precision. It is common for the evaluator to fear low precision in a tool, even when high recall is more important than high precision, because the lower the precision, the more false positives there will be in the tool's output to be vetted, to the dismay of the human vetters. The tool developer wonders what is the proper balance between recall and precision for the tool, and would benefit from guidance in finding the proper balance. *[Principal Ideas]* This article reviews the measures used to evaluate such a tool: such as recall; precision; the F-measure; $\lambda$, the frequency of true positives in the tool's input; selectivity, the ratio of the size of the tool's output to the size of the tool's input; and summarization. It derives mathematically a formula relating a tool's recall, precision, $\lambda$, and selectivity that shows mathematically that the lower $\lambda$ is, the lower precision can be before the tool's selectivity, and thus the tool's output, become too big for comfortable vetting. *[Contribution]* With this formula, it is possible to show mathematically that, e.g.,
if

$\lambda$ is 10% or less

and

the tool's recall is 100%,

then

the tool's precision can be as low as 50%,

and

the size of the tool's output,

the file to vet to weed out false positives,

is still no bigger than double the number of true positives in the input, and thus the output.

Such a conclusion is useful for knowing how much precision can be traded away in the pursuit of as much recall as possible.

**Keywords:** *F*-measure, Frequency of true positives, Hairy task, Natural-Language processing, NLP, Precision, Recall, Selectivity, Summarization, Tool, Vetting, Weeding out false positives

# 1 Introduction

## 1.1 Background and Assumptions for a Scenario

Here are the background and assumptions for a common scenario in the development and evaluation of a tool, $t$[1], that performs the task, $T$. Here, $t$ is based on natural-language (NL) processing (NLP) [14, 43] or on artificial intelligence (AI) or a large language model (LLM) [48, 53]; $T$ is a hairy task that classifies examined items as meeting the task's criteria or not; and $t$ tries to identify and return *all and only* those items of $t$'s input that $T$ classifies as meeting $T$'s criteria. As a result, $t$ is evaluated mainly in terms of two measures[2], *recall* and *precision*. Many add the harmonic mean, $F_1$, of recall and precision that weights them equally [49].

$T$ is *hairy* [3, 4] if and only if

1. $T$ involves *understanding* the entirety of its input,
2. $T$ is non-algorithmic[3],
3. humans do the understanding that $T$ requires with no particular difficulty, all the time, on small input, but
4. humans find doing $T$ on large input to be unmanageable.

For the first three criteria, there are no measures. For the fourth, there is a *vague* (in the linguistics sense) measure, $\lambda$, the frequency within the input of the items that $T$ classifies as meeting $T$'s criteria. Basically, the smaller $\lambda$ is, the more unmanageable $T$ is when the input is large. The measure is vague, because in any situation, it is not clear what value of $\lambda$ is the boundary between manageable and unmanageable. These properties of a hairy task are what make the existence of a tool to do that task so attractive [5].

In the following bulleted list of examples of hairy tasks, immediately after each item, which itself is a list of specific tasks, comes a list of indications of domains in which the tasks of the item are relevant. The meanings of these indications are:

**RE**: Requirements Engineering
**SE**: Software Engineering
**La**: Law
**Md**: Medicine
**PS**: Privacy and Security
**NL**: Natural-Language Processing
**IR**: Information Retrieval
**AI**: Artificial Intelligence

Here, AI includes learned machines (LMs), artificial neural networks (ANNs), and large language models (LLMs).

Examples of hairy tasks include

1. searching for ambiguities and other defects in NL [**RE**, **La**, **NL**] [7, 12, 46, 50],
2. searching for abstractions, features, or bug reports in NL [**RE**, **SE**] [17, 19, 31],
3. mapping regulations into requirements for a CBS that implements the regulations [**RE**, **La**, **PS**] [6, 28],
4. searching for trace links among software-development artifacts [**RE**, **SE**] [6, 9, 20, 33, 39],
5. searching for defects in code and other software-development artifacts [**SE**] [36]
6. searching an artifact library for reusable artifacts that are relevant to a given issue [**SE**, **RE**] [27, 32],
7. electronic discovery in legal proceedings [**La**] [10, 18, 41],

[1] Apologies for using what look like mathematical formulae in narrative text. The text resulting from a first attempt to write this paper using only words in the narrative text resulted in ambiguous text. For example, it became difficult to distinguish a tool in general from the specific running example tool, $t$, that is the subject of the running scenario, the figures, and the formulae. Does the term "the tool" refer to the previously introduced tool in general or the running example tool? Thus, "$t$" is used even in the narrative to talk about the specific running example tool, to reserve "tool" to talk about tasks in general. For a list of all identifications used throughout this article, see Appendix A.

[2] For now, it suffices to say that $t$'s *recall* is the percentage of the input items that meet $T$'s classification that $t$ returns, i.e., measuring the all-ness of $t$, and $t$'s *precision* is the percentage of the input items that $t$ returns that meet $T$'s classification i.e., measuring the only-ness of $t$, Section 1.4 gives complete definitions of these measures and other concepts.

[3] If $T$ were algorithmic, we would simply implement one of the algorithms into $t$, and $t$ would have 100% recall and 100% precision, with no tradeoff necessary.

8. searching for regulations, statutes, and court decisions that are relevant to a given issue [**La**] [15, 29, 35],
9. systematic review in evidence-based medicine [**Md**] [18, 41],
10. the creation of fully labeled test collections for information retrieval evaluation and machine learning [**NL**, **IR**, **AI**] [18, 41],
11. searching for signs of disease in images [**Md**] [30, 40, 45],
12. fraud detection, customer segmentation, medical diagnosis, and sentiment analysis [**RE**, **Md**, **PS**] [42], and
13. identifying, by careful search, examination, or probing, the existence or presence of [something], e.g., detecting cybersecurity threats [**PS**, **IR**, **AI**] [45].

Each of these tasks may be carried out using NLP, IR, or AI techniques. Here again, AIs include LMs, ANNs, and LLMs. Clearly, the tradeoff between recall and precision occurs in a lot more than in just RE and SE.

To be clear, there are many tasks, and tools, for which this article is irrelevant. For sure, many a task is *not* hairy:

- doing the task does not require understanding the entirety of its input,
- the task is algorithmic with guaranteed $100\%$ recall and $100\%$ precision, or
- the frequency of true positives in the task's input is large.

Even for a hairy task,

- for which $100\%$ recall is desired, not always is doing the task *necessary*; it can be optional.
- not always is $100\%$ recall needed; a lower recall suffices.
- not always is recall more important than precision; e.g., for a search engine, avoiding false positives is critical to not scaring customers away.

This article offers nothing to the developers of these other kinds of tasks.

### 1.2 The Scenario

In a typical scenario that you might undergo in the development of $t$ for $T$, $T$ is both hairy and critical enough that $100\%$ recall is essential. Moreover, there is no avoiding doing $T$, no matter how tedious it is. Finally, the time constraints placed on completing $T$ are generally generous enough to allow $T$ to be done as well as possible, including to allow concurrent or consecutive performance of $T$ by different actors.

You have determined that the typical input to which $T$ is applied has a $\lambda$ of $10\%$. Thus, fully $90\%$ of the items examined in any performance of $T$ will not be what $T$ is searching for, making $T$ very tedious. This tediousness will discourage most from doing $T$ well, finishing $T$ once started, or doing $T$. You recognize that the very tediousness of $T$ vitiates against achieving the necessary $100\%$ recall. To alleviate the tediousness, and to try to achieve the necessary $100\%$ recall, you decide to develop a tool, $t$, to perform $T$ [5].

In preparation for building a gold set (a.k.a. "ground truth") input with which to evaluate the effectiveness of any tool in performing $T$ [49], you remember that the typical input to which $T$ is applied has a $\lambda$ of $10\%$. Therefore, you construct a representative gold set input, $\mathsf{G}$, so that a randomly distributed $10\%$ of the searchable items in $\mathsf{G}$ are true positives, intended to be precisely the items found when performing $T$ or running any tool for $T$ on $\mathsf{G}$.

You have even used the process of construction of the gold set to determine that the average domain-expert human being that is on the committee performing $T$ manually on $\mathsf{G}$ achieves $85\%$ recall [3, 4]. This $85\%$ is designated to be the HAR (humanly achievable recall) of $T$. Consequently, you would be happy with any tool that achieves any recall greater than $85\%$, but you will still work hard to optimize $t$'s recall, often by trading away some of $t$'s precision. You have learned that for most classification tasks, recall and precision trade off. If, however, $t$'s recall ends up being enough[4] less than $85\%$, $T$ will have to be done manually to get as much recall as is possible.

You have developed $t$ for $T$ and worked very hard to try to get $t$ to achieve $100\%$ recall. No matter what $t$'s recall is, say $X\%$, you expect that $t$ will return some false positives, i.e., will have a precision of less than $100\%$. You know that humans will have to vet the tool's output to weed out these false positives, to leave only the $X\%$ of the true positives[5]. That no arbitrary time constraints were placed on the performance of $T$ permits vetting as a means to

[4] How much is enough depends on the criticality of $T$ in the context in which $T$ is performed.

[5] This is assuming that vetting does not inadvertently weed out any true positives. Actually, vetting almost always reduces recall to slightly less than $X\%$.

achieve higher precision than achieved by the raw $t$. Nevertheless, you hope that the precision is not very much less than $100\%$, because the lower the precision, the more to vet for false positives, and the more to vet, the more tedious vetting becomes, approaching the tedium of a manual search of the entire input to $t$.

For a complete discussion of the general tradeoff, see the paper titled "Evaluation of Tools for Hairy Requirements Engineering Tasks" (ETHRET) [3].

You run $t$ on G, and are pleasantly shocked that $t$'s recall is $100\%$. Your $t$ found *every* true positive in G! Wow!!!!! Your elation comes to a sudden end when you discover that $t$'s precision is *only* $50\%$, giving an $F_1$ of *only* $66.7\%$! "Oh no!" you think. "This is a case of the useless $100\%$-recall tool, that does no searching and just returns the entire input." Such a tool is useless because vetting the tool's output is the *same* as doing $T$ manually on the tool's input, which is what you developed $t$ to avoid having to do! The reality is that $t$ is much better than the useless tool, but you are not convinced.

While building $t$, you had constructed another tool, $t'$, for $T$, whose recall was only $90\%$, not as good as that of $t$, but still better than the HAR for $T$, $85\%$. Better yet, $t'$'s precision is $70\%$, significantly better than $t$'s abysmal $50\%$!

Everything you have been taught over the years says that even when nominally, very high recall, at or near $100\%$, is required, low precision is very, very bad, because human beings will *refuse* to vet the output, because vetting the output will be the same as the dreaded manual search of the input. You have been taught to play a compromise game, to somehow balance recall and precision. So, like everyone else, you calculate the $F_1$ measure that weights recall and precision equally:

- For $t$, with $100\%$ recall and $50\%$ precision, $F_1$ is $66.67\%$, and
- for $t'$, with $90\%$ recall and $70\%$ precision, $F_1$ is $78.75\%$.

Finally, you declare $t'$ to be the better tool overall. Yes, $t'$'s recall is not as good as $t$'s, but $t'$'s recall is still better than the HAR, *and, most importantly*, its precision is *much better than* $t$'s. At least the vetters are appeased and will be happy!

While it is impossible to know what really went through the minds of the developers of tools reported in the literature, at least the reported results of many papers about tools for hairy tasks show some evidence of similar kinds of thinking [2, 8, 23, 24, 31, 37, 39, 51, e.g.].

But Whoa! You really wish there were some rational way, based on data about the context of $t$ and $T$, that could inform this decision, and ultimately, the underlying tradeoff. This decision, based on recall and precision or $F_1$ is not using any measure of what is really relevant, namely the size of $t$'s output to be vetted. The fear of a low precision or low $F_1$ arises from the desire to avoid having to vet a large output from $t$, especially one whose size approaches that of $t$'s input. Then vetting $t$'s output becomes the same as doing $T$ manually on $t$'s input. Clearly, the simplest thing to do is to just determine the size of $t$'s output. If this size is small enough that the vetters do not complain, then the precision does not matter.

For reasons explained in Section 3.3, it is sometimes premature or not possible to calculate the size of $t$'s output. Also, in evaluating the effectiveness of $t$ in general, the issue is *not* the raw size of $t$'s output in any one application of $t$, but the size of $t$'s output as a function of the size of $t$'s input, e.g., the ratio of the size of $t$'s output to the size of $t$'s input. Let's see what we can learn about this ratio from $T$'s $\lambda$ and $t$'s recall and precision.

Consider $t$, with $100\%$ recall and $50\%$ precision, and $t'$, with $90\%$ recall and $70\%$ precision. Taking into account both recall and precision, the size of the output, to be vetted, of a run of $t$ on an input is $156\%$ of the size of the output of a run of $t'$ on the same input. That is, $t'$ requires $36\%$ less vetting than does $t$, and the vetting of $t'$'s output will find that only $30\%$ of the output is false positives, and not $50\%$, as with $t$'s output.

However, the fact that $\lambda = 10\%$ allows pinning down a number of properties of $t$'s and $t'$'s outputs that effect the ease of vetting. By drawing diagrams similar to those in Figure 3, but with actual data, you are able to see that

- for $t$, with $100\%$ recall and $50\%$ precision,
  the size of $t$'s output to vet
  is only $20\%$ of the size of $t$'s input,
  that would have to be manually searched
  if there were no tool,

  and
- for $t'$, with $90\%$ recall and $70\%$ precision,
  the size of $t'$'s output to vet

is only $12.9\%$ of the size of $t$'s input,
that would have to be manually searched
if there were no tool.

It seems to me that getting an additional $10\%$ of recall when running $t$ on an input instead of running $t'$ is worth the additional $7.1\%$-of-the-size-of-an-input that needs to be vetted when running $t$ on the input instead running $t'$. This is the case especially in a context in which $100\%$ recall is required and in which the alternative to having a decent tool for $T$ is manually searching the entire input.

All of these specific data can be generalized into a mathematically-derived formula that relates four values, $T$'s $\lambda$, $t$'s recall, $t$'s precision, and $t$'s selectivity, i.e., $t$'s ratio of the size of $t$'s output for an input to the size of that input. This formula allows computing any one of the values from the other three.

The formula shows that, for example,
if

$\lambda$, the frequency of true positives in $t$'s input, is $10\%$

and

$t$'s recall is $100\%$,

then

$t$'s precision can be as low as $50\%$,

and

the size of $t$'s output,
the file to vet to weed out false positives,
is still no bigger than double the number of true positives in the input, and thus the output.

This observation leads to the concept of a *maximum tolerable* size of $t$'s output. While you are designing $t$ for $T$, learn from the people that will be vetting $t$'s output the maximum size of $t$'s output, expressed as a percentage of the size of $t$'s input, that they are willing to vet without complaint, under the constraints that

- $T$ must be done;
- $t$ must achieve a recall of at least $T$'s HAR; and
- if $t$ does not achieve a recall of at least $T$'s HAR, then $T$ must be done manually.

The formulae for a tool's recall and precision are such that if you manage to optimize $t$ to achieve $100\%$ recall, once you know $T$'s $\lambda$ and the maximum tolerable size of $t$'s output, expressed as a percentage of the size of $t$'s input, you can calculate $t$'s minimum tolerable precision. Moreover, the smaller $T$'s $\lambda$ is, the smaller $t$'s minimum tolerable precision will be. For example, if $\lambda = 10\%$ and the maximum tolerable size of $t$'s output is $20\%$, then $t$'s minimum tolerable precision is $50\%$. It gets better with a smaller $\lambda$; if $\lambda = 1\%$ and the maximum tolerable size of $t$'s output is $20\%$, then $t$'s minimum tolerable precision is $5\%$.

Wow![6]

The remainder of this Introduction section first describes the context of the problem and gives the definitions needed to describe the problem and the eventual solution. Then, it outlines the rest of the article.

[6] The reality of the formulae is that if you have managed to optimize $t$ to achieve $100\%$ recall, once you know $T$'s $\lambda$ and $t$'s minimum tolerable precision you can calculate the maximum tolerable size of $t$'s output. expressed as a percentage of the size of $t$'s input. That is, the formulae are bidirectional. However, it's hard to imagine a scenario in which one would use the formulae in this direction. To a human being, precision is just a number, but the size of $t$'s output determines the amount of vetting that needs to be done. If the human desires to minimize the size of $t$'s output, to minimize the tedium of vetting, then "the maximum tolerable size of $t$'s output" expresses a real constraint that the human being may have.

### 1.3 Context

The collection of artifacts examined by $T$ forms $T$'s *input*, $\mathsf{I}$. $T(\mathsf{I}) = \mathsf{Q}$ is the result of performing $T$ on $\mathsf{I}$. Additionally, $t(\mathsf{I}) = \mathsf{O}$ is the output of *running* $t$ on $\mathsf{I}$. Ideally, $t(\mathsf{I}) = T(\mathsf{I})$, but this equality is rarely the case[7].

Because of the criticality of the computer-based system (CBS) for whose development $T$ is being done, e.g., the CBS has safety or security concerns, there is no option of simply not doing $T$. $T$ must be done *completely*, finding everything that $T$ seeks, achieving as close as possible to a perfect job, of finding all and only what $T$ seeks, namely the $\mathsf{Q}$ mentioned above. The alternative to using $t$, or any other tool, to do $T$ is to perform $T$ entirely manually.

### 1.4 Definitional Background

This subsection defines the terms needed to state the problem and its solution. This article builds on and extends my previous work on evaluation of tools for hairy tasks [3, 4] and uses vocabulary from this work verbatim.

#### 1.4.1 Answers

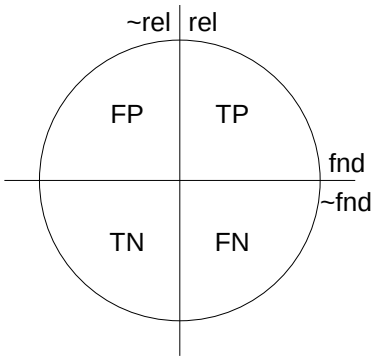


**Fig. 1.** The Universe of an NLP Tool

An *answer* is a unit of whatever $T$ examines in $\mathsf{I}$, in an attempt to find all relevant answers in $\mathsf{I}$. $T$ tries to distinguish between

- *relevant* answers and
- *not relevant* or *irrelevant* answers.

A relevant answer often called a "correct answer", but "correct" is heavily overloaded term. This article leaves "correct" for its everyday meaning.

#### 1.4.2 Role of $t$

The tool, $t$, tries to *find* all and only relevant answers in $\mathsf{I}$, leaving as *not found* all and only irrelevant answers in $\mathsf{I}$, but it generally cannot do so perfectly.

Figure 1 shows the universe of $t$ for $T$. Running $t$ partitions the space of answers into 4 regions:

- $\mathsf{TP}$, the set of true positives (TPs), the relevant answers that are found,
- $\mathsf{FN}$, the set of false negatives (FNs), the relevant answers that are not found,
- $\mathsf{TN}$, the set of true negatives (TNs), the irrelevant answers that are not found, and
- $\mathsf{FP}$, the set of false positives (FPs), the irrelevant answers that are found.

[7] The evidence for this claim is that in the literature, a tool for a hairy task that has 100% recall is rare, and among the rare cases of reporting the HAR for a task, an HAR of 100% is rare.

### 1.4.3 Confusion Matrix

From an input $\mathsf{I}$, $t$ produces an output $t(\mathsf{I}) = \mathsf{O} = \mathsf{fnd} = (\mathsf{TP} \cup \mathsf{FP})$, while $\sim \mathsf{fnd} = (\mathsf{TN} \cup \mathsf{FN})$. The sizes (numbers of answers) of the 4 regions are the values of the 4 cells of a *confusion matrix*,

| $\lvert\mathsf{FP}\rvert$ | $\lvert\mathsf{TP}\rvert$ |
|---|---|
| $\lvert\mathsf{TN}\rvert$ | $\lvert\mathsf{FN}\rvert$ |

.

### 1.4.4 Vetting

*Vetting* is process of a human's examining the elements of $\mathsf{O}$ from $t$, to partition it into its 2 subsets, $\mathsf{TP}$ and $\mathsf{FP}$, such that

$$\mathsf{O} = \mathsf{fnd} = (\mathsf{TP} \cup \mathsf{FP}) \text{ and } \emptyset = (\mathsf{TP} \cap \mathsf{FP}).$$

The basic manual examination of $\mathsf{I}$ that occurs during the performance of $T$ on $\mathsf{I}$ can be considered a vetting of $\mathsf{I}$. Nevertheless, "vetting" is reserved to describe examining $\mathsf{O} = t(\mathsf{I})$ for its FPs, and "examining", "finding", or "searching" is used to describe a completely manual performance of $T$ that $t$ tries to imitate.

### 1.4.5 The Important $\lambda$

An important number is $\lambda$:

$$\lambda = \frac{\lvert\mathsf{rel}\rvert}{\lvert\mathsf{I}\rvert}.$$

We can use $\lambda$ as a measure of hairiness; the smaller $\lambda$ is for $T$, the hairier $T$ is, because the smaller $\lambda$ is for $T$, the more answers have to be examined to find *one* TP.

### 1.4.6 Define $R$ and $P$

$R$ and $P$ are defined:

$$R = \frac{\lvert\mathsf{fnd} \cap \mathsf{rel}\rvert}{\lvert\mathsf{rel}\rvert} = \frac{\lvert\mathsf{TP}\rvert}{\lvert\mathsf{TP}\rvert + \lvert\mathsf{FN}\rvert},$$

and

$$P = \frac{\lvert\mathsf{fnd} \cap \mathsf{rel}\rvert}{\lvert\mathsf{fnd}\rvert} = \frac{\lvert\mathsf{TP}\rvert}{\lvert\mathsf{TP}\rvert + \lvert\mathsf{FP}\rvert}.$$

In any case, $\min(P) = \lambda = \dfrac{\lvert\mathsf{rel}\rvert}{\lvert\mathsf{I}\rvert}$.

### 1.4.7 Define $F$

A composite of $R$ and $P$, often called "correctness", is $F$:

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

the harmonic mean of $R$ and $P$.

### 1.4.8 Weighted $F$

When $R$ and $P$ are not equally important, e.g., when the cost of a FN is more than the cost of a FP, i.e., when $\lambda$ is small and it's easy to dismiss a presented FP, there's a weighted $F$:

$$F_\beta \ = \ (1+\beta^2) \times \frac{P \times R}{(\beta^2 \times P) + R}$$

Even with $\beta = 5$, $P$ is largely irrelevant to $F_\beta$. Later, we'll see that maybe $\beta$ should not be squared in the formula for $F_\beta$, as it normally is.

### 1.4.9 What Should $\beta$ Be?

Assuming that vetting an answer in $\mathsf{O}$ costs the same as examining an answer in $\mathsf{I}$, one reasonable estimate for $\beta$ is that $\beta = \frac{1}{\lambda}$, because, on average, finding one relevant answer will require examining $\frac{1}{\lambda}$ answers in $\mathsf{I}$. Typically, vetting an answer in $\mathsf{O}$ is faster than examining an answer in $\mathsf{I}$ [11, 34, 44]. So, the $\beta = \frac{1}{\lambda}$ estimate is conservative; it is smaller than it could be. An important datum about $t$ is the ratio, **a**, of

(the time to decide relevance of one answer in manually doing $T$ on $\mathsf{I}$)
to
(the time to decide relevance of one answer in manually doing $T$ on $\mathsf{O} = t(\mathsf{I})$).

**a** is called *vetting speed-up*[8] of $t$ for $T$. Conveniently, $\mathbf{a} > 1$ if vetting an answer in $\mathsf{O}$ is faster than examining an answer in $\mathsf{I}$, just one more reason to not care about low $P$. In fact, I'm beginning to think that we should define $\beta = \frac{1}{\lambda} \times \mathbf{a}$.

### 1.4.10 Trading $R$ and $P$

Generally, we get higher $R$ at the expense of lower $P$, and vice versa. The extremes of this tradeoff are:

- $t$ returns the entire $\mathsf{I}$, $t(\mathsf{I}) = \mathsf{O}$: $R = 100\%$ and $P = \lambda = \min(P)$, and
- $t$ returns only one relevant answer, $t(\mathsf{I}) = \mathsf{O}$ and $|\mathsf{O}| = 1$ and $\mathsf{O} \subset \mathsf{rel}$: $P = 100\%$ and $R = \frac{1}{|\mathsf{rel}|}$.

These extremes are useless, because in either case, the entire $\mathsf{I}$ has to be manually searched in order to find the relevant answers. This extreme tradeoff means that care must be taken when $t$'s $R$ is $100\%$ or nearby. Is $t$ really only a useless tool or very close to one? In Figure 2, Part (a) shows the configuration of the output for a useless tool, and Part (b) shows the configuration of the output for a close to useless tool. In each configuration diagram, the space of answers is divided into regions, corresponding to two or more of the quadrants of the universe of an NLP tool shown in Figure 1. A solid line represents a boundary that is visible to any human who has run the tool on input $\mathsf{I}$ to get $\mathsf{O}$. A dotted line represents a boundary that is invisible to this human. For example, in Part (b), the human can see which of the answers in $\mathsf{I}$ are in $\mathsf{O}$. However, the human knows only that $\mathsf{O}$ consists of a mix of TPs and FPs, but there is no clear separation between them. Instances of each are scattered randomly throughout $\mathsf{O}$.

Fortunately, many an algorithm with high $R$ and low $P$ returns an $\mathsf{O}$ enough smaller than $\mathsf{I}$, that manual vetting of $\mathsf{O}$ takes less time than manual examination of $\mathsf{I}$, from both

- the vetting speed-up, and
- that $\mathsf{O} = t(\mathsf{I})$ is significantly smaller than $\mathsf{I}$.

We can get out of the $R = 1$ useless extreme if $t(\mathsf{I}) = \mathsf{O}$ is significantly smaller than $\mathsf{I}$, so that vetting $\mathsf{O}$ is much faster than examining $\mathsf{I}$.

[8] Think: "speed-up" is roughly "acceleration"; hence "**a**".

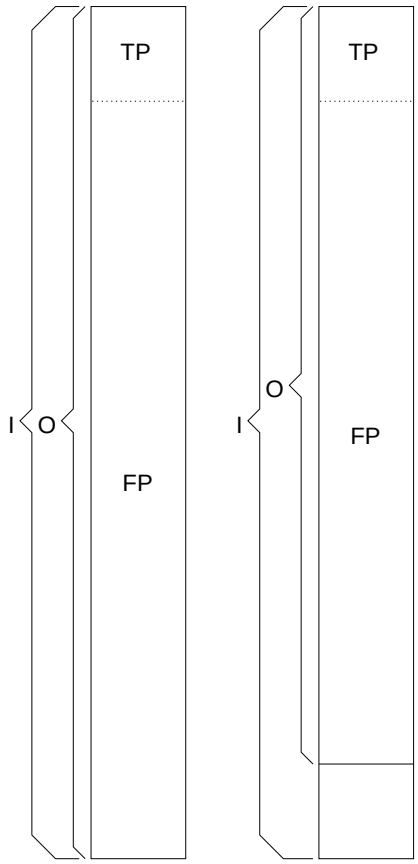


**Fig. 2.** Different Configurations of Tool Outputs

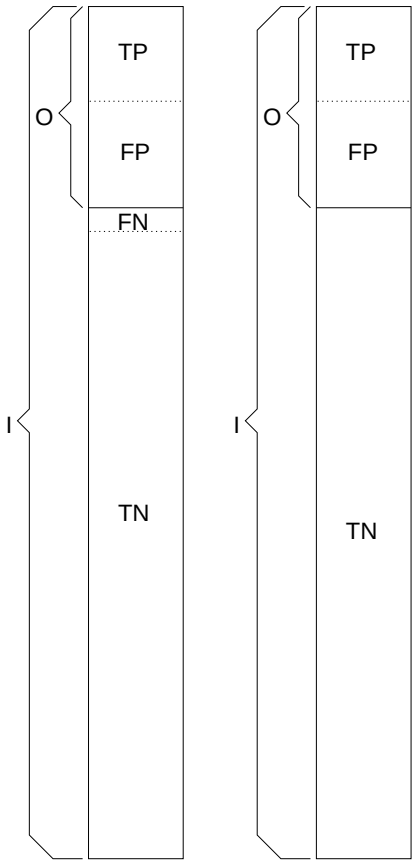


**Fig. 3.** More Different Configurations of Tool Outputs

#### 1.4.11 Define $S$ and $s$

Summarization [3], $S$, and selectivity [22], $s$, are two measures of the amount by which $|\mathsf{O}| = |t(\mathsf{I})| < |\mathsf{I}|$.

$$S = \frac{|\sim \mathsf{fnd}|}{|\sim \mathsf{fnd} \cup \mathsf{fnd}|} = \frac{|\sim \mathsf{fnd}|}{|\sim \mathsf{rel} \cup \mathsf{rel}|} = \frac{|\mathsf{TN}| + |\mathsf{FN}|}{|\mathsf{TN}| + |\mathsf{FN}| + |\mathsf{TP}| + |\mathsf{FP}|}.$$

$$s = \frac{|\mathsf{fnd}|}{|\sim \mathsf{fnd} \cup \mathsf{fnd}|} = \frac{|\mathsf{fnd}|}{|\sim \mathsf{rel} \cup \mathsf{rel}|} = \frac{|\mathsf{TP}| + |\mathsf{FP}|}{|\mathsf{TN}| + |\mathsf{FN}| + |\mathsf{TP}| + |\mathsf{FP}|} = 1 - S.$$

We would like $R = 1$, $\lambda$ small, and $s$ just bigger than $\lambda$. Then we do not care if $P$ is low! There's a relationship between $R$, $P$, $s$, and $\lambda$ that we can exploit.

### 1.5 Related Work

There are descriptions of related work scattered throughout this article, each occurring in the part of this article that is naturally related to the related work. These descriptions occur in Sections 1.1, 1.2, 2, 3.2, and 3.3.

Even in the work introducing, using, and reporting selectivity and stressing the importance of low selectivity to ensure the manageability and tolerability of vetting a tool's output for false positives [21, 44, 44], I have not seen any explicit statement of the relation between $\lambda$ and selectivity captured in the formula and its derived equations.

Hayes *et al* describe the importance of a small $s$ in making vetting manageable [22], but they neither consider nor express the derived relationship between $\lambda$, $P$, and $s$. They operate in a domain in which $R$ as low as 60% is deemed tolerable. Moreover, they do not consider trading low $P$ to try to achieve $R = 1$.

I did hint at the relationship in Section 5.1 of ETHRET [3] when I showed a relationship between summarization, $S = 1 - s$, and $P$ that used $1 - \lambda$. Even knowing this relationship, the formula derived in this article seemed to me to be new. Also, even as late as 2024 and 2025, some 3 and 4 years after the publication of ETHRET, one finds evaluations, such as by Hey and two different sets of co-authors [23, 24], still using $F_1$ as the basis to choose the best of several tools for a task, without consideration of the size of the output to be vetted. Evidently, others besides me have not understood the $S$-measure-based explanation of Section 5.1 of ETHRET.

### 1.6 The Rest of this Article

In the rest of this article, Section 2 introduces the common problem concerning the tradeoff between $R$ and $P$ that occurs during the evaluation of tools for hairy tasks. Section 3 derives a formula that shows the reality of this tradeoff, and explains why the formula is useful, even though knowing just the value of one variable is enough. Section 4 provides a simplified formula for when $R = 1$, and gives (1) a plot of the value of this simplified formula for each of three illustrative configurations of values of its two variables and (2) a table of the values of this simplified formula for $9 \times 5$ key configurations of values of its two variables. Section 5 offers some thoughts about improving the definition of the weighted $F$-measure, and Section 6 concludes the article.

## 2 The Problem

This section generalizes and summarizes the scenario of Section 1 into a general statement of a common problem in the development and evaluation of a tool, $t$ for a hairy task, $T$. In this situation, most recognize that the most important measure the evaluation of $t$ is $R$ [2, 8, 22, 33]. However, high $R$ often comes at the cost of low $P$. Tool developers appear to be afraid of anything smacking of low $P$ [31, 37].

Indeed, when Hey, Keim, and Corallo evaluated several tools for the hairy task of trace link recovery, they chose the best based on values of $F_1$, and explained why [24] [all emphases are mine]:

> Therefore, we also provide precision, recall, and $\mathrm{F}_1$-score that focus on the occurrences only. We choose $\mathrm{F}_1$-score instead of a different F-measure, as for our following task of *filtering irrelevant requirements elements* both recall and precision are equally important. When valuing recall higher than precision, a lower precision *could result* in not being able to filter any requirements elements as *too many* are considered for example functional. If we value precision higher than recall, a lower recall may result in actually relevant elements being filtered as they are regarded as, e.g., having no functional aspects.

Essentially, the reason for not valuing recall higher than precision focuses on concerns about the "following task of filtering irrelevant requirements elements" i.e., the vetting for false positives. They worry that that low precision *could* — not will — result in there being "too many" false positives to do an accurate filtering job.

They are correct that low precision *could* result in having to vet an output that is almost the same size as as the original input. However, it is clear that the data they have collected to be able to calculate the recall and precision values that they show would permit them to determine if, in fact, there *are* too many false positives to do an accurate filtering job.

Some tool developers appear to go through heroic efforts to increase $P$, even at the cost of lowering $R$ [20, 39]. The main sign of this fear of low $P$ is the the continued insistence on using $F_1$, which weights $R$ equally with $P$, with no discussion of $\lambda$ or of the size of $\mathsf{O}$ in comparison to the size of $\mathsf{I}$, *sometimes even* in the presence of data, e.g., the number of TPs in the gold set of reported size, that show that $\lambda$ for $T$ is small, in the order of magnitude of $0.1$ [6, 16, 25, 26, 38, 47, 52].

This fear arises from two factors,

1. getting low $P$ means that $\mathsf{O}$ must be vetted manually to weed out the FPs that have crept in with the TPs contributing to the high $R$, and
2. the extremes of the tradeoff between $R$ and $P$, in which getting either 100% $R$ or 100% $P$ means that the vetting amounts to doing the whole task manually anyway.

The developers of $t$ fear that if there are too many FPs in $\mathsf{O}$, the human vetters will be so frustrated from doing what they perceive as effectively doing $T$ on close to the whole bloody $\mathsf{I}$, something that the use of the $t$ is supposed to avoid [20].

However, this fear is somewhat irrational, because the alternative to using $t$ in this high criticality case is that $T$ *must be* done on the *whole bloody* $\mathsf{I}$. At least, the vetting is done on *only* $\mathsf{O}$, which is often significantly smaller than $\mathsf{I}$ [1, 22]. Moreover, $t$ can present each item of $\mathsf{O}$ in a format that facilitates deciding whether the item is a TP or an FP, e.g., for a trace link, the output might show directly the artifacts connected by the link. Vetting a link in $\mathsf{O}$ will be significantly easier than *searching* for a link and its connected artifacts in situ in $\mathsf{I}$, i.e., in this case, $\mathbf{a} > 1$ [11, 26, 34, 37, 39].

Even so, the fear borne of the extreme tradeoff overrides rationality, and many evaluations weight $R$ and $P$ equally, implicitly by using $F_1$. However, for a good $t$, the tradeoff is far from the extreme. A good $t$ *does* make the $\mathsf{O}$ to vet significantly smaller than $\mathsf{I}$. That is, $t$ has small $s$, so that low $P$ can be tolerated because the size of the $\mathsf{O}$ to vet is significantly smaller than the $\mathsf{I}$ that would have to be examined manually if $t$ were not available.

The developers of $t$ wonder what is the proper balance between recall and precision for $t$, wish for guidance in finding the proper balance. The purpose of this article is to provide this guidance in the form of mathematically derived formulae based on the definitions of $R$, $P$, and standard measures of properties of the input, that quantify just how low $t$'s $P$ can go before vetting will become intolerable.

## 3 Reality of Tradeoff

Part (a) of Figure 3 shows the desired configuration of $O = t(I)$. While the figure purports to illustrate all configurations in which $\lambda$ is small, and $s$ is small, it actually shows the configuration when $R = 0.8$, $P = 0.5$, $\lambda = 0.125$, and $s = 0.2$.

### 3.1 Formulae

Always,

$$P \times |\mathsf{O}| = |\mathsf{TP}| = R \times |\mathsf{rel}|.$$

Thus, $\mathsf{rel} = \dfrac{P \times |\mathsf{O}|}{R}$.

But, $\lambda = \dfrac{|\mathsf{rel}|}{|\mathsf{I}|}$.

So, $\lambda = \dfrac{P \times |\mathsf{O}|}{R \times |\mathsf{I}|}$.

Then, $\lambda \times \dfrac{R}{P} = \dfrac{|\mathsf{O}|}{|\mathsf{I}|}$, and $\dfrac{|\mathsf{O}|}{|\mathsf{I}|} = s$.

Thus,

$$s = \lambda \times \frac{R}{P}.$$

This equation is hereinafter known as "the formula". From the formula, three other equations can be derived:

$$\lambda = s \times \frac{P}{R}, P = \lambda \times \frac{R}{s}, \text{ and } R = P \times \frac{s}{\lambda}.$$

So, given any three of $R$ of $t$, $P$ of $t$, $s$ of $t$, and $\lambda$ of $T$, the fourth can be calculated.

### 3.2 Sanity Checks

This section provides two sanity checks on the formula, (1) mathematical and (2) empirical.

The mathematical check verifies that the formula,

$$s = \lambda \times \frac{R}{P} = \lambda \times R \times \frac{1}{P},$$

is consistent with the definitions of its terms. Remember from the universe of an NLP tool diagram in Figure 1 that $|\mathsf{rel}| = |\mathsf{TP}| + |\mathsf{FN}|$, and observe that $|\mathsf{TN}| + |\mathsf{FN}| + |\mathsf{TP}| + |\mathsf{FP}| = |\mathsf{I}|$. Then, using the definitions of $s$, $\lambda$, $R$, and $P$ in Section 1.4, $s = \lambda \times R \times \dfrac{1}{P}$ can be rewritten as

$$\frac{|\mathsf{TP}| + |\mathsf{FP}|}{|\mathsf{I}|} = \frac{|\mathsf{TP}| + |\mathsf{FN}|}{|\mathsf{I}|} \times \frac{|\mathsf{TP}|}{|\mathsf{TP}| + |\mathsf{FN}|} \times \frac{|\mathsf{TP}| + |\mathsf{FP}|}{|\mathsf{TP}|}.$$

If, in the right-hand-side product of the rewriting, you mutually cancel the green $|\mathsf{TP}| + |\mathsf{FN}|$s in one numerator and in one denominator, and you mutually cancel the red $|\mathsf{TP}|$s in one numerator and in one denominator, you are left with $|\mathsf{TP}| + |\mathsf{FP}|$ in the numerator and with $|\mathsf{I}|$ in the denominator. This is *exactly* the left hand side of the equation.

The empirical check verifies that data from a past evaluation of tools for the hairy task of tracing satisfies the formula via one of its equations, $\lambda = s \times \dfrac{P}{R}$. Sundaram, Hayes, and Dekhtyar (SHD) experimentally evaluated a variety of IR methods applied to the task of tracing of open source datasets [44]. SHD do not use any F-measure, instead couching their conclusions strictly on the basis of recall, precision, and selectivity. They make tradeoffs consistent with the importance of recall over precision for the tracing task, and they take into account low selectivity in choosing the winning method. SHD give in their Table 1, the precision, recall, and selectivity of the variety of IR methods applied to the MODIS and CM-1 datasets. From data that SHD give describing the MODIS and CM-1 datasets, it is possible to estimate $\lambda$ for each dataset [3, Section 4.1]. The MODIS dataset consists of 19 high-level requirements and 49 low-level requirements, and has 41 true links, each from a high-level requirement to a low-level requirement. The CS-1 dataset consists of 235 high-level requirements and 220 design elements, and has 361 true links, each from a high-level requirement to a design element. Thus,

$$\lambda_{\text{MODIS}} = \frac{41}{19 \times 49} = \frac{41}{931} = 0.044, \text{ and } \lambda_{\text{CM}-1} = \frac{361}{235 \times 220} = \frac{361}{51700} = 0.007.$$

For the MODIS dataset, the tf-idf+TH method achieves $R = 100\%$, $P = 10.1\%$, and $s = 43.1\%$[9]. Therefore,

[9] I picked the tf-idf+TH method for both datasets, because for the MODIS dataset, the method achieves a perfect recall with an abysmal precision, and for the CM-1 dataset, the recall is almost as good and the precision is worse! These values, thus, provide a good test of this article's claim.

$$\lambda = s \times \frac{P}{R} = .431 \times \frac{.101}{1.0} = 0.044!$$

For the CM-1 dataset, the tf-idf+TH method achieves $R = 97.1\%$, $P = 1.5\%$, and $s = 42.8\%$. Therefore,

$$\lambda = s \times \frac{P}{R} = .428 \times \frac{.015}{.971} = 0.007!$$

For each dataset, the two ways to calculate $\lambda$ agree exactly after rounding in the third decimal place. These calculations show that the method described in Section 4.1 of ETHRET [3] of estimating $\lambda$ from the data is correct, as it produces values for $\lambda$ that work in the formula and its equations!

### 3.3 Why All This Bother?

Why bother with the formula and its derived equations, when all you need to do is calculate $s$ to determine how much smaller the job of vetting O will be than the job of examining I? The main reason to bother is to allow identification of legacy tools for hairy tasks whose incomplete evaluations led to their being thought to be worse than they actually are.

Many an old paper about tools for a hairy task reports only $R$, $P$, often $F_1$, and less often $F_2$, in addition to or instead of $F_1$, for the tools. It may choose the tools with the highest $F$-measure, rather than with the highest $R$ as the best [31, 39] or be less than excited about the tools because the values of $P$ for the tools are not high enough even though the values of $R$ are very good, even occasionally $100\%$ [2, 8, 37, 51]. We know that the task that the tools do has a small $\lambda$, e.g., less than $0.1$. Even better, the paper may quote data that allow estimating $\lambda$. If not, for a well-known task, such as tracing, an estimate of $\lambda$ derived from other tool evaluations can be used. We want to know if it is worthwhile to re-evaluate the tools on the basis of more complete data [3]. It is worthwhile if given the available data, $s$, can be shown to be small enough to satisfy the vetters.

Appendices B through E show the calculations for four papers published in the past. For the first three, the calculations show that re-evaluation of the studied tools is in order, while for the last, re-evaluation of the studied tool is probably a waste of effort.

The second reason to bother is that the formula and its equations help the designer of a tool for a task with a known $\lambda$ and a target $s$, to know just how much $P$ can be traded away in a quest to maximize $R$. The next section considers this tradeoff by first simplifying the formula and the equations under the assumption that $R = 1$, the highest possible recall for any tool. Then, for any fixed $\lambda$, it becomes possible to plot the formula as a two-dimensional diagram to *visualize* the tradeoff between $s$ and $P$. The section shows plots of $P$ as a function of $s$ for three specific $\lambda$s, and it provides a table of the value of $P$ for nine key values of $s$ and five key values of $\lambda$.

## 4 Reality of Tradeoff When $R = 1$

Part (b) of Figure 3 shows the desired configuration of $O = t(I)$. While the figure purports to illustrate all configurations in which $R = 1$, $\lambda$ is small, and $s$ is small, it actually shows the configuration when $R = 1$, $\lambda = 0.1$, and $s = 2 \times \lambda$, and $P = 50\%$.

### 4.1 Formulae

Since our goal for $t$ is $100\%$ $R$, Then, the formula simplifies to:

$$s = \frac{\lambda}{P}.$$

The three other equations simplify to:

$$\lambda = s \times P,\ P = \frac{\lambda}{s},\ \text{and}\ 1 = P \times \frac{s}{\lambda}.$$

So, for $R = 1$, given any two of $P$ of $t$, $s$ of $t$, and $\lambda$ of $T$, the third can be calculated. The last of the equations can be used as a sanity check in any situation.

Suppose that you are willing to tolerate vetting an $\mathsf{O}$ that is no bigger than $s \times |\mathsf{I}|$, for $0 \leq s < 1$. Then, you can tolerate a $P$ no lower than $\frac{\lambda}{s}$. The smaller $\lambda$ is, the lower a tolerable $P$ is.

It is ironic that the very property of the input of a hairy task — a small $\lambda$ — that makes having a tool for the task so attractive [5], that requires the tool to have as close to $100\%$ recall as possible [3], is what makes low precision tolerable, up to a $\lambda$-dependent limit.

### 4.2 Plots of $P$ Versus $s$ for Some Key $\lambda$s When $R = 1$

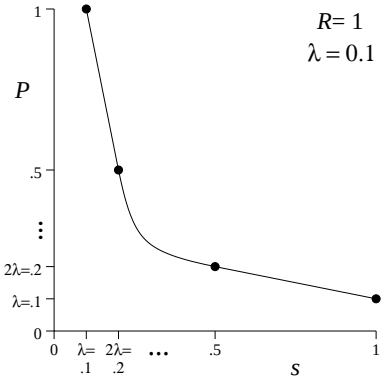


**Fig. 4.** Plot $P$ vs $s$ for $\lambda = 0.1$

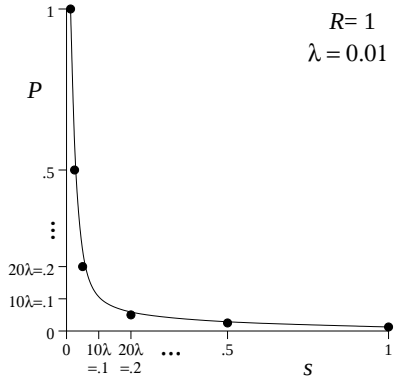


**Fig. 5.** Plot $P$ vs $s$ for $\lambda = 0.01$

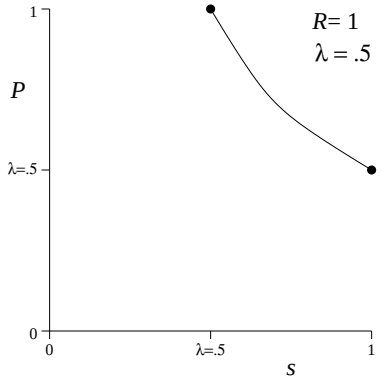


**Fig. 6.** Plot $P$ vs $s$ for $\lambda = 0.5$

Figures 4 through 6 show plots of $P$ as a function of $s$ for some key $\lambda$s, namely, $0.1$, $0.01$, and $0.5$, a small, a very small, and a large $\lambda$, all when $R = 1$. Notice the initial rapid decline of $P$ with $\lambda = 0.1$. Notice the even more rapid initial rapid decline of $P$ with $\lambda = 0.01$. However, with $\lambda = 0.5$, the decline of $P$ is almost a $45°$ straight line segment from $(0.5, 1)$ to $(1, 0.5)$.

Figures 2 and 3 in Section 5.1 of ETHRET [3] hints at the relationship that these plots show. However, the relationship is not as clear because it is couched in terms of $S$ rather than $s$, causing the relevant plot to be concave downward and leftward ($\urcorner$) rather than concave upward and rightward ($\llcorner$). So, the rapid drop off of the value of $P$ against increasing $s$ for small $\lambda$s is not at all clear.

### 4.3 Table of Minimum Tolerable $P$s When $R = 1$

Table 1 shows the minimum tolerable $P$ for some values of $s$ and some values of $\lambda$ when $R = 1$. The table shows clearly that the smaller $\lambda$ is, for any particular $s$, the lower the $P$ you can tolerate. The first row shows the useless $R = 100\%$ case, in which $|\mathsf{O}| = |\mathsf{I}|$. In this case $P = \lambda$ identically. The formula calculates some meaningless $P$ values that are greater than $1.0$. Each is for an $s$ value that is less than $\lambda$, an impossibility. Each of these meaningless $P$ values is colored red. The table shows clearly that the lower the $P$ you want to tolerate, for any particular $s$, the smaller $\lambda$ must be. Equivalently, the smaller $\lambda$ is, for any particular $P$, the larger the $s$ you can tolerate.

## 5 Rethinking Definition of $F_\beta$

Some [2, 8] define the weighted $F$-measure as

$$F_\beta \;=\; (1 + \beta) \times \frac{P \times R}{(\beta \times P) + R}$$

instead of as

**Table 1.** Minimum Tolerable $P$s For Some $\lambda$s and Some $s$s When $R = 1$

| | | $\lambda$ | | | | |
|---|---|---|---|---|---|---|
| | | 0.5 | 0.2 | 0.1 | 0.05 | 0.01 |
| $s$ | $\frac{1}{1} = 1.000$ | 0.5 | 0.2 | 0.1 | 0.05 | 0.01 |
| | $\frac{3}{4} = 0.750$ | 0.667 | 0.267 | 0.133 | 0.067 | 0.013 |
| | $\frac{1}{2} = 0.500$ | 1.0 | 0.4 | 0.2 | 0.10 | 0.02 |
| | $\frac{1}{3} = 0.333$ | 1.5 | 0.6 | 0.3 | 0.15 | 0.03 |
| | $\frac{1}{4} = 0.250$ | 2.0 | 0.8 | 0.4 | 0.20 | 0.04 |
| | $\frac{1}{5} = 0.200$ | 2.5 | 1.0 | 0.5 | 0.25 | 0.05 |
| | $\frac{1}{6} = 0.167$ | 3.0 | 1.2 | 0.6 | 0.30 | 0.06 |
| | $\frac{1}{8} = 0.125$ | 4.0 | 1.6 | 0.8 | 0.40 | 0.08 |
| | $\frac{1}{10} = 0.100$ | 5.0 | 2.0 | 1.0 | 0.50 | 0.10 |

$$F_\beta \;=\; (1+\beta^2) \times \frac{P \times R}{(\beta^2 \times P) + R}.$$

That is, they use just $\beta$ rather than $\beta^2$. While, I have never seen an explanation for using $\beta$ instead of $\beta^2$ or even vice versa, it seems clear that if you want to use $F_\beta$ as a signal of $\lambda = \dfrac{1}{\beta}$ in its role of indicating how tolerable a low $P$ is, then using $\beta^2$ gives too much power to $R$ over $P$ in $F_\beta$, and instead, just $\beta$ should be used.

## 6 Conclusion

In the past, evaluators of tools for hairy RE tasks have struggled to find the correct tradeoff between $R$ and $P$, given the contexts of the tools' use. This article provides some definitions and a formula that allow the tradeoff to be conducted rationally on the basis of easily collected data about the inputs to the tools. It simplifies the formula for the target case of 100% recall and provides plots and tables illustrating the simplified formula's behavior for key settings of its domain values. The formulae, the plots, and tables should help the developers of tools for hairy task develop tools better suited for the context of their usage than in the past. In the end, the very property of the input of a hairy task — a small $\lambda$, a small frequency of true positives in the input — that makes having a tool for the task so attractive, that requires the tool to have as close to 100% recall as possible, is what makes low precision, up to a $\lambda$-dependent limit, tolerable.

## Acknowledgements

Daniel Berry's work was supported in part by a Canadian NSERC grant NSERC-RGPIN227055-22. **Data Availability Statement**: This article deals with no data at all.

## References

1. Antoniol, G., Canfora, G., Casazza, G., De Lucia, A., Merlo, E.: Recovering traceability links between code and documentation. IEEE Transactions on Software Engineering **28**(10), 970–983 (2002). https://doi.org/10.1109/TSE.2002.1041053
2. Arora, C., Sabetzadeh, M., Briand, L., Zimmer, F.: Automated checking of conformance to requirements templates using natural language processing. IEEE Transactions on Software Engineering **41**(10), 944–968 (2015). https://doi.org/10.1109/TSE.2015.2428709

3. Berry, D.M.: Evaluation of tools for hairy requirements engineering tasks. Empirical Software Engineering **26**(111), 1–77 (2021). https://doi.org/10.1007/s10664-021-09986-0
4. Berry, D.M.: Empirical evaluation of tools for hairy natural language requirements engineering tasks. In: Ferrari, A., Ginde, G. (eds.) Natural Language Processing for Requirements Engineering, pp. 381–405. Springer, Cham (2025). https://doi.org/10.1007/978-3-031-73143-3_14F
5. Berry, D.M., Gacitua, R., Sawyer, P., Tjong, S.F.: The case for dumb requirements engineering tools. In: Proceedings of the International Working Conference on Requirements Engineering: Foundation of Software Quality (REFSQ). pp. 211–217 (2012). https://doi.org/10.1007/978-3-642-28714-5_18
6. Breaux, T.D., Gordon, D.G.: Regulatory requirements traceability and analysis using semi-formal specifications. In: Proceedings of the International Working Conference on Requirements Engineering: Foundation of Software Quality (REFSQ). pp. 141–157 (2013). https://doi.org/10.1007/978-3-642-37422-7_11
7. Chantree, F., Nuseibeh, B., de Roeck, A., Willis, A.: Identifying nocuous ambiguities in natural language requirements. In: Proceedings of the IEEE International Requirements Engineering Conference (RE). pp. 56–65 (2006). https://doi.org/10.1109/RE.2006.31
8. Cleland-Huang, J., Czauderna, A., Gibiec, M., Emenecker, J.: A machine learning approach for tracing regulatory codes to product specific requirements. In: Proceedings of the ACM/IEEE International Conference on Software Engineering (ICSE). pp. 155–164 (2010). https://doi.org/10.1145/1806799.1806825
9. Cleland-Huang, J., Zemont, G., Lukasik, W.: A heterogeneous solution for improving the return on investment of requirements traceability. In: Proceedings of the IEEE International Requirements Engineering Conference (RE). pp. 230–239 (2004). https://doi.org/10.1109/ICRE.2004.1335680
10. Cormack, G.V., Grossman, M.R., Harbison, A., O'Halloran, T., McManus, B.: Unbiased validation of technology-assisted review for ediscovery. In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR). pp. 2677–2681 (2024). https://doi.org/10.1145/3626772.3657903
11. Delater, A., Paech, B.: Tracing requirements and source code during software development: An empirical study. In: Proceedings of the International Symposium on Empirical Software Engineering and Measurement (ESEM). pp. 25–34 (2013). https://doi.org/10.1109/ESEM.2013.16
12. Fabbrini, F., Fusani, M., Gnesi, S., Lami, G.: An automatic quality evaluation for natural language requirements. In: Proceedings of the International Working Conference on Requirements Engineering: Foundation of Software Quality (REFSQ). pp. 1–18 (2001), https://refsq.upc.edu/2001/papers/p3.pdf
13. Fazelnia, M., Koscinski, V., Herzog, S., Mirakhorli, M.: Lessons from the use of natural language inference (NLI) in requirements engineering tasks. In: Proceedings of the IEEE International Requirements Engineering Conference (RE). pp. 103–115. IEEE Computer Society, Los Alamitos, CA, USA (2024). https://doi.org/10.1109/RE59067.2024.00020
14. Ferrari, A., Ginde, G.: Natural Language Processing for Requirements Engineering. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-73143-3
15. Gartner: Legal research software reviews and ratings (2026), https://www.gartner.com/reviews/market/legal-research-software
16. Gervasi, V., Zowghi, D.: Supporting traceability through affinity mining. In: Proceedings of the IEEE International Requirements Engineering Conference (RE). pp. 143–152 (2014). https://doi.org/10.1109/RE.2014.6912256
17. Goldin, L., Berry, D.M.: AbstFinder: A prototype abstraction finder for natural language text for use in requirements elicitation. Automated Software Engineering **4**, 375–412 (1997). https://doi.org/10.1023/A:1008617922496
18. Grossman, M.R., Cormack, G.V., Roegiest, A.: TREC 2016 total recall track overview (2016), http://trec.nist.gov/pubs/trec25/trec2016.html
19. Guzman, E., Maalej, W.: How do users like this feature? A fine grained sentiment analysis of app reviews. In: Proceedings of the IEEE International Requirements Engineering Conference (RE). pp. 153–162 (2014). https://doi.org/10.1109/RE.2014.6912257
20. Hayes, J.H., Dekhtyar, A., Osborne, J.: Improving requirements tracing via information retrieval. In: Proceedings of the IEEE International Requirements Engineering Conference (RE). pp. 138–147 (2003). https://doi.org/10.1109/ICRE.2003.1232745
21. Hayes, J.H., Dekhtyar, A., Larsen, J., Guéhéneuc, Y.G.: Effective use of analysts' effort in automated tracing. Requirements Engineering Journal **23**(1), 119–143 (2018). https://doi.org/10.1007/s00766-016-0260-8
22. Hayes, J.H., Dekhtyar, A., Sundaram, S.K.: Advancing candidate link generation for requirements tracing: The study of methods. IEEE Transactions on Software Engineering **32**(1), 4–19 (2006). https://doi.org/10.1109/TSE.2006.3
23. Hey, T., Fuchß, D., Keim, J., Koziolek, A.: Requirements traceability link recovery via retrieval-augmented generation. In: Hess, A., Susi, A. (eds.) Proceedings of the International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ). pp. 381–397 (2025). https://doi.org/10.1007/978-3-031-88531-0_27
24. Hey, T., Keim, J., Corallo, S.: Requirements classification for traceability link recovery. In: Proceedings of the IEEE International Requirements Engineering Conference (RE). pp. 155–167 (2024). https://doi.org/10.1109/RE59067.2024.00024
25. Hippargi, V., Kamsties, E., Naumann, J.: Evaluating the capabilities of LLMs in traceability maintenance for automotive system and software requirements. In: Joint Proceedings of REFSQ 2025 Workshops, Doctoral Symposium, Posters & Tools Track, and Education and Training Track. p. Short Paper 2 (2025), https://ceur-ws.org/Vol-3959/NLP4RE-short2.pdf

26. Hübner, P., Paech, B.: Using interaction data for continuous creation of trace links between source code and requirements in issue tracking systems. In: Proceedings of the International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ). pp. 291–307 (2017). https://doi.org/10.1007/978-3-319-54045-0_21
27. Isakowitz, T., Kauffman, R.: Supporting search for reusable software objects. IEEE Transactions on Software Engineering **22**(6), 407–423 (1996). https://doi.org/10.1109/32.508314
28. Kosenkov, O., Elahidoost, P., Gorschek, T., Fischbach, J., Méndez, D., Unterkalmsteiner, M., Fucci, D., Mohanani, R.: Systematic mapping study on requirements engineering for regulatory compliance of software systems. Information and Software Technology **178**, 107622 (2025). https://doi.org/10.1016/j.infsof.2024.107622
29. LexisNexis: Lexis advance Quicklaw (2026), https://www.lexisnexis.com/en-ca/products/lexis-advance-quicklaw-overview
30. Lu, J., Tong, X., Wu, H., Liu, Y., Ouyang, H., Zeng, Q.: Image classification and auxiliary diagnosis system for hyperpigmented skin diseases based on deep learning. Heliyon **9**(9), e20186 (2023). https://doi.org/10.1016/j.heliyon.2023.e20186
31. Maalej, W., Kurtanović, Z., Nabil, H., Stanik, C.: On the automatic classification of app reviews. Requirements Engineering Journal **21**(3), 311–331 (2016). https://doi.org/10.1007/s00766-016-0251-9
32. Maarek, Y., Berry, D., Kaiser, G.: An information retrieval approach for automatically constructing software libraries. IEEE Transactions on Software Engineering **17**(8), 800–813 (1991). https://doi.org/10.1109/32.83915
33. Mahmoud, A., Niu, N.: Supporting requirements to code traceability through refactoring. Requirements Engineering Journal **19**(3), 309–329 (2014). https://doi.org/10.1109/RE.2013.6636703
34. Maro, S., Steghöfer, J., Hayes, J., Cleland-Huang, J., Staron, M.: Vetting automatically generated trace links: What information is useful to human analysts? In: Proceedings of the IEEE International Requirements Engineering Conference (RE). pp. 52–63 (2018). https://doi.org/10.1109/RE.2018.00-52
35. Matich, T., Lenon, J.: 9 best legal research resources: Databases, tools and software. Clio (2025), https://www.clio.com/blog/best-legal-research-tools/
36. Menzies, T., Dekhtyar, A., Distefano, J., Greenwald, J.: Problems with precision: A response to "comments on 'data mining static code attributes to learn defect predictors'". IEEE Transactions on Software Enginerring **33**(9), 637–640 (2007). https://doi.org/10.1109/TSE.2007.70721
37. Merten, T., Krämer, D., Mager, B., Schell, P., Bürsner, S., Paech, B.: Do information retrieval algorithms for automated traceability perform effectively on issue tracking system data? In: Proceedings of the International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ). pp. 45–62 (2016). https://doi.org/10.1007/978-3-319-30282-9_4
38. Patel, K., Modi, H., Shah, U.: LACE-HC: A lightweight attention-based classifier for efficient hierarchical classification of software requirements. In: Proceedings of the International Working Conference on Requirements Engineering: Foundation of Software Quality (REFSQ). pp. 181–196 (2025). https://doi.org/10.1007/978-3-031-88531-0_13
39. Quirchmayr, T., Paech, B., Kohl, R., Karey, H.: 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). pp. 255–272 (2017). https://doi.org/10.1007/s10664-018-9597-6
40. Radcliffe, C., Grant, M.: Infectious disease images: A remarkable, free resource. Open Forum Infectious Diseases **8**(12), ofab560 (2021). https://doi.org/10.1093/ofid/ofab560
41. Roegiest, A., Cormack, G.V., Grossman, M.R., Clarke, C.L.: TREC 2015 total recall track overview (2016), http://trec.nist.gov/pubs/trec24/trec2015.html
42. Ross, L.: What is AI classification in machine learning? (2026), https://bigid.com/blog/ai-classification/, big·id
43. Ryan, K.: The role of natural language in requirements engineering. In: Proceedings of the IEEE International Symposium on Requirements Engineering (ISRE). pp. 240–242 (1993). https://doi.org/10.1109/ISRE.1993.324852
44. Sundaram, S.K., Hayes, J.H., Dekhtyar, A.: Baselines in requirements tracing. In: Proceedings of the Workshop on Predictor Models in Software Engineering (PROMISE). pp. 1–6 (2005). https://doi.org/10.1145/1082983.1083169
45. Theofanos, M., Choong, Y.Y., Jensen, T.: AI use taxonomy. Tech. Rep. NIST AI 200-1, National Institute of Standards and Technology, U.S. Department of Commerce (2024). https://doi.org/10.6028/NIST.AI.200-1, NIST Trustworthy and Responsible AI
46. Tjong, S.F., Berry, D.M.: 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). pp. 80–95 (2013). https://doi.org/10.1007/978-3-642-37422-7_6
47. Uyar, E.B., Gürsoy, A.E., Gökçe, C., Tuğba, Temizel, T.: Low-level hardware requirement classification using large language models: Challenges, insights, and future directions for embedded control systems. In: Joint Proceedings of REFSQ 2025 Workshops, Doctoral Symposium, Posters & Tools Track, and Education and Training Track. p. Paper 3 (2025), https://ceur-ws.org/Vol-3959/NLP4RE-paper3.pdf
48. Vogelsang, A., Fischbach, J.: Using large language models for natural language processing tasks in requirements engineering: A systematic guideline. In: Ferrari, A., Ginde, G. (eds.) Natural Language Processing for Requirements Engineering, pp. 435–456. Springer, Cham (2025). https://doi.org/10.1007/78-3-031-73143-3_16
49. Wikipedia: Precision and recall (viewed May 2026), https://en.wikipedia.org/wiki/Precision_and_recall

50. Wilson, W.M., Rosenberg, L.H., Hyatt, L.E.: Automated analysis of requirement specifications. In: Proceedings of the ACM/IEEE International Conference on Software Engineering (ICSE). pp. 161–171 (1997). https://doi.org/10.1145/253228.253258
51. Yang, H., Roeck, A.N.D., Gervasi, V., Willis, A., Nuseibeh, B.: Analysing anaphoric ambiguity in natural language requirements. Requirements Engineering Journal **16**(3), 163–189 (2011). https://doi.org/10.1007/s00766-011-0119-y
52. Zhang, Z., Liang, B., Wong, K.F.: LSRM: A hybrid LLM-SBERT approach for mapping user requirements to product functionalities in complex products. In: Proceedings of the IEEE International Requirements Engineering Conference (RE). pp. 267–279 (2025). https://doi.org/10.1109/RE63999.2025.00033
53. Zhao, L., Alhoshan, W.: Machine learning for requirements classification. In: Ferrari, A., Ginde, G. (eds.) Natural Language Processing for Requirements Engineering, pp. 19–59. Springer, Cham (2025). https://doi.org/10.1007/78-3-031-73143-3_2

## A List of Identifications

This appendix lists all the identifications, to have them in one place for easy reference by the reader.

First is a list of artifacts

| Artifact | Explanation |
|---|---|
| $T$ | hairy task |
| $t$ | tool for $T$ |
| G | gold set document for $T$ |
| I | input to which $T$ or $t$ is applied |
| Q | output from applying $T$ to I |
| O | output from applying $t$ to I |
| ETHRET | "Evaluation of Tools for Hairy Requirements Engineering Tasks" [3] |

Next comes a list of measures.

| Measure | Explanation |
|---|---|
| $R$ | recall of $t$ |
| $P$ | precision of $t$ |
| $S$ | summarization of $t$ |
| $s$ | selectivity of $t$ |
| **a** | vetting speedup (acceleration) for $t$ |
| $\lambda$ | frequency of relevant answers among answers in I for $T$ |
| $\beta$ | weight of recall over precision for $T$ |
| HAR | humanly achievable recall for $T$ |

## B Hey, Keim, and Corallo, 2024

Recall from Section 2, that Hey, Keim, and Corallo (HKC) evaluated several tools for the hairy task of trace link recovery (TLR), and they chose the best based on only the values of $F_1$ [24].

Let us get an upper bound on the value of $s$ by taking the highest recall and lowest precision that HKC show together with the largest $\lambda$ this article uses for the tracing task. Their Table IV shows 50 different recall values and 50 different precision values.

> The maximum recall value among the 50 is $1.00$.
> The minimum precision value among the 50 is $0.017$.
> The second lowest precision value among the 50 is $0.363$,
> the third lowest is $0.445$, and
> the maximum is $0.968$.

Clearly, the $0.017$ is an outlier, as the $49$ other precision values lie well distrubuted between $0.363$ and $0.968$. So let us use $0.363$ as the minimum precision value to calculate $s$. With a maximum $R = 1.00$, a minimum $P = .363$, and a minimum $\lambda = .054$, the maximum $s = 0.149$. Note that if the minimum $P = .017$ is used, the maximum $s = 3.18$,

an impossibility. If HKC's filterers can tolerate an output whose size is not more than $15\%$ of that of the input as an alternative to having to manually filter the input, then HKK should consider reevaluating their tools on the assumption that recall is at least $18.40$ times as important as precision.

HKC do evaluate also a fully automated tool for which there will be no filtering task. In this case, recall and precision *are* of equal weight. However, given that the HARs that they report are all greater than $.851$ and that the highest recall for any fully automated tool is $.532$, it appears to me unconscionable to use their fully automated tool in a high criticality development.

## C Fazelnia, Koscinski, Herzog, and Mirakhorli, 2024

Fazelnia, Koscinski, Herzog, and Mirakhorli (FKHM) investigate the use of a new technique, natural-language inference (NLI) to automate several RE tasks, including classification of requirements, identification of requirements specification defects, and detection of conflicts in stakeholders' requirements [13], that are known to be hairy. They compare the effectiveness of NLI with classical NLP methods, prompt-based methods, conventional transfer learning, LLM-powered chatbots, and probabilistic methods on these RE tasks. Their experiments, conducted in a variety of learning settings, demonstrate conclusively that their NLI method achieves higher $F_1$ values than any of the other methods for these RE tasks.

However, they use $F_1$, which assumes equal importance to recall and precision, with no examination of the relative importance of recall and precision in the context in which their tasks are applied.

Let us see what conclusion they would come to if they used recall instead of $F_1$ to choose the best method and Let us get an upper bound on the value of $s$ for some of their methods applied to two of their tasks.

FKHM say that the value that they report for any measure, $R$, $P$, or $F_1$, is a weighted mean of all the individual values of the measure, each weighted by its sample size. I could not be certain that I could apply the sample sizes reported in the paper correctly. Using the data in their Table IX for the ChatGPT, NoRBERT, Prompt-Based, NLI, and Indicator Term Freq. methods doing the classification task, I calculated just the raw, unweighted mean of $F_1$ for the NLI method applied to the classification task as $83.73\%$, which is very close to their reported weighted mean of $83\%$ for the same. Consequently, I use the raw, unweighted mean of the appropriate values in the appropriate table in all the calculations below.

In their Table IX about the requirements classification task, among the ChatGPT, NoRBERT, Prompt-Based, NLI, and Indicator Term Freq. methods, NLI achieves the highest unweighted mean $F_1$ of $83.73\%$. However, among the same methods, ChatGPT achieves the highest unweighted mean $R$ of $89.64\%$, while NLI achieves only $78.91\%$. ChatGPT achieves an unweighted mean $P$ of $58.18\%$, which might be considered alarmingly low. Assuming a $\lambda$ of the task of $10\%$, based on my gut feeling informed by my experience, the maximum $s$ is $15.4\%$. If this estimated $\lambda$ is correct, the size of output to be vetted is only $1.54$ times the number of TPs in the input and $1.71$ times the number of TPs in the output.

In their Table X about the task to detect defects in a requirements specification, among the ChatGPT, NoRBERT, Prompt-Based, and NLI methods, NLI achieves the highest unweighted mean $F_1$ of $81.33\%$. For this task, among the same methods, NLI achieves the highest unweighted mean $R$ of $93.33\%$, while NoRBERT achieves only $81.33\%$, as the second highest. So, for this task, $F_1$ happens to choose the same method as does $R$. NLI achieves an unweighted mean $P$ of $74.00\%$, which would be considered fairly decent. Assuming a $\lambda$ of the task of $20\%$, based on my gut feeling informed by my experience, the maximum $s$ is $25.22\%$. If this estimated $\lambda$ is correct, the size of output to be vetted is only $1.26$ times the number of TPs in the input and $1.40$ times the number of TPs in the output.

Thus, it is probably worth for FKHM to reevaluate the methods to come to more nuanced and stronger conclusions.

## D Hey, Fuchß, Keim, and Koziolek, 2025

Hey, Fuchß, Keim, and Koziolek (HFKK) evaluate several methods to recover trace links with the help of retrieval-augmented generation (RAG) [23]. They report, "Although there are no statistically significant differences between the prompt types, it is noticeable that if precision is as important as recall [[10]] (F1-score), chain-of-thought [(CoT)] prompting can improve the performance on inter-requirements traceability link recovery." Among the datasets used

[10] This article effectively claims that the antecedent of this vacuously true conditional is false for hairy tasks.

to test the methods are the MODIS and CM-1 datasets used in SHD's evaluation discussed in Section 3.2. SHD's evaluation provides the data for calculate $\lambda$ values to use with HFKK's $R$ and $P$ values to calculate $s$ values. For the other datasets, e.g., the WARC dataset, we will use $\lambda = \frac{1}{18.40} = 0.054$, the largest, and therefore most conservative, $\lambda$ obtained from a variety of evaluations of tracing tools reported in Section 4.1 of ETHRET [3].

For the CM-1 dataset, maximizing $F_1$ chooses CoT, CPT-4o, but maximizing $R$ chooses any KISS variant, even though their $P$ values are lower than for the $F_1$-chosen approach. Taking the *lowest* $P$ value, the worst case, gives $R = 64.4\%$ and $P = 33.0\%$, and using $\lambda_{\mathrm{CM}-1} = .007$ from before, gives $s = 1.37\%$, a small enough $s$.

For the MODIS dataset, maximizing $F_1$ chooses KISS, Code-llama, but maximizing $R$ chooses any KISS variant, even though all but one of their $P$ values are lower than for the $F_1$-chosen approach, which is again a KISS variant. Taking the *lowest* $P$ value, the worst case, gives $R = 26.8\%$ and $P = 14.5\%$, and using $\lambda_{\mathrm{MODIS}} = .044$ from before, gives $s = 8.1\%$, a small enough $s$.

For the WARC dataset, maximizing $F_1$ chooses CoT, CPT-4o, but maximizing $R$ chooses the first three KISS variants, even though their $P$ values are lower than for the $F_1$-chosen approach. Taking the *lowest* $P$ value, the worst case, gives $R = 69.1\%$ and $P = 37.3\%$, and using $\lambda = .054$ from reports of a variety of tracing tools, gives $s = 10.0\%$, a small enough $s$.

In each case, the $s$ is small enough that the the vetting job is significantly smaller than the full manual search. Thus, in each case, it is worth to evaluate the tools again with the full set of data recommended by Berry [3, 4].

## E Merten, Krämer, Mager, Schell, Bürsner, and Paech, 2016

Merten, Krämer, Mager, Schell, Bürsner, and Paech (MKMSBP) evaluated the effectiveness of five different IR algorithms for the task of extracting trace links for a CBS from the data in the issue tacking system (ITS) for the CBS's development [37]. One particular algorithm achieved $R = 1.0$, but they dismissed it because its $P = 0.02$. Their paper did not provide enough data to estimate $\lambda$.

Using $\lambda = 0.054$, the largest $\lambda$ obtained from a variety of evaluations of tracing tools reported in Section 4.1 of ETHRET [3], with $R = 1.0$ and $P = 0.02$ gives $s = 2.7$ which is impossible.

The next question is "What must $\lambda$ be to get $s \leq 1$ when $R = 1.0$ and $P = 0.02$?" The equation, $\lambda = s \times P$, says that $\lambda$ must be less than $0.2$, i.e., smaller than 1 in 50.

The "50" here is what ETHRET calls $\beta$, which is $\frac{1}{\lambda}$. ETHRET reports $\beta$ values for the tracing task from various sources as 23.17, 22.70, 143.21, 23.65, 27.91, 57.05, and 18.40. This article has been using the minimum of these $\beta$ values, 18.40 to obtain its largest estimated $\lambda = 0.054$ for the tracing task. Only two of the seven reported $\beta$ values, 143.21 and 57.05, are greater than equal to 50. The larger of these two, 143.21 implies $\lambda = .007$, which in turn, with $R = 1.0$ and $P = 0.02$ gives $s = 0.35$, Thus, with an estimated $\lambda$ that is likely too small, the output of the tool to be vetted is no smaller than an uncomfortably big 35% of the tool's input.

It's probably not worth the effort for MKMSBP to re-evaluate their tool.