

Clone Detector Evaluation Can Be Improved: Ideas from Information Retrieval

Andrew Walenstein and Arun Lakhotia
Software Research Laboratory
Center for Advanced Computer Science
University of Louisiana at Lafayette
E-mail: walenste@ieee.org

Abstract

We believe the techniques for evaluating clone detectors can be improved, and that the improvements can lead the way to better clone detector research and research results. Current techniques are based on simple performance measures borrowed from information retrieval (IR) research. Here we argue that additional IR evaluation measures can be usefully imported.

1. Limitations of current research

Automated methods for finding software clones have, in the past, been primarily evaluated according to simple measures adapted from prior work in the field of *information retrieval* (IR). The main stated purpose of an IR system is to enable users to find *documents* which are *relevant* to the user based on some approximation of their needs in the form of a *query*. One classic way of evaluating IR systems is to examine their scores on *precision* and *recall*. These measures are usually neatly defined and related by way of a table (see e.g., van Rijsbergen [5]):

	RELEVANT	IRRELEVANT	
RETRIEVED	$A \cap B$	$\bar{A} \cap B$	B
NOT RETRIEVED	$A \cap \bar{B}$	$\bar{A} \cap \bar{B}$	\bar{B}
	A	\bar{A}	

Precision and recall are then defined as

$$precision = \frac{|A \cap B|}{|B|}, \quad recall = \frac{|A \cap B|}{|A|}$$

Ideally, an IR system would yield simultaneously high recall and high precision: it would return all the relevant documents and *only* the relevant documents.

Application of these ideas to clone detection (CD) are typified by Kontogiannis' evaluation [3]. In this formulation, clone pairs correspond to documents, and relevance is

interpreted to mean a clone pair is relevant if and only if it is really a clone pair. Effectively there is but a single query being evaluated, namely: "find all clones" [3]. Normally the result is in the form of a simple set. That is, the clone candidates are unweighted and order is considered unimportant. Perfect recall, in this context, means that all clones are found; perfect precision means no false clones are reported.

This basic evaluation template has been followed by several researchers, and precision and recall values have become the *de facto* basis for empirical evaluation of automated CD systems [1–4]. Although this has proven to be a useful tactic, it limits what can be said about the CD results and how they may be compared. We feel that it worthwhile to expand on these evaluation techniques, and that exploring other IR-based measures techniques is a good first step. IR has developed an arsenal, and from examining them several useful directions for CD evaluation might be derived.

2. Evaluation measures and techniques

IR evaluation methods are generally based on some simple measure of the query result set. Although precision and recall appear to be the main measures, another relatively common measure is called *fallout*. Referring to the table from Section 1, it is defined as $fallout = |\bar{A} \cap B|/|\bar{A}|$. Fallout thus measures the fraction of results which are false positives. We are not aware of any CD studies directly measuring fallout, yet it seems to be a measure we should be paying closer attention to. It is clearly relevant when applying clone detectors in automation contexts such as in automated code refactoring. There, a low fallout maps directly to the ideal of using *conservative* analyses. In interactive applications, fallout neatly captures the intuitive measure of how much useless rubbish the user will have to slog through to find the truly relevant clones.

If detector A has higher recall but lower precision than B, which is better? One way of resolving this conundrum in IR is to introduce a "composite" measure, which provides a total ordering on the results. For instance, one might

compare the ratio of precision to recall, or precision to fall-out [5]. Although composite measures can be contentious, using a composite measure will, at the very least, appropriately confine disagreements to the measure’s definition.

Another possibly fruitful avenue is to compare *precision/recall curves* [5]. A precision-recall curve can illustrate how a detector responds to changes in its tunable parameters. Many detectors provide one or more tunable parameters as a threshold value for making clone comparisons. For instance, the detector used by Kontogiannis [3] provided a setting for how many insertions or deletions were allowed before two candidates were considered non-clones. If one systematically varies such a threshold value, one may obtain the precision/recall curve, which is more informative than precision and recall values measured for an *ad hoc* tuning value. With such curves one may establish and compare the useful ranges of detectors. For instance, how does detector A’s precision differ from B’s when the recall is set to 80%? 50%? 30%? Such curves might be used to help evaluators empirically pick appropriate threshold values, and could yield better advice for choosing CD systems to meet specific task demands.

3. Multiple queries

IR evaluation is founded on being able to specify multiple queries. This makes sense because users will, in general, encounter many different task contexts. Assuming a space of queries allows one to define useful notions such as “average precision” and “average recall” [5]. These measures may be of more service than measures for the single, generic query of “find all clones”. It would be difficult to imagine cases in IR where the query is fixed, and it seems clear to us that the CD field must step beyond performance comparisons on merely the “find all clones” query. For instance, consider the case where a developer creates a clone clique by cloning a function four times over, after which another developer made a bug fix to one of them, but not the others. It is likely that these other clones also needed to be updated, and that these related clones may cause maintenance headaches. A “find parallel maintenance headaches” query would be desirable. CD systems need to be evaluated against these and other queries.

4. Ranked results

One vigorous area of research in IR has been the investigation of ranked query result sets. Anyone using WWW search tools knows about ranked results: the most “interesting” results are presented first. Many existing CDs are easily modified so that they generate ranked outputs instead of amorphous result sets: merely rank the clone candidates

according to the “closeness” metric being used. Then CDs can be evaluated not only on the overall content of the result sets, but also according to how they are ranked so that the user is presented the most salient clone candidates first. However this is but the first step in an intriguing research path. The ranking generally needs to reflect the user’s goals. Consider the previously mentioned task of finding inconsistent parallel modifications. The clones within such a clone clique are bound to be different due to the inconsistent fixes. Yet if the detectors rank candidates by how similar they are to one another, then such problem clones might rank well down the list and exact clones causing no problems would come first. It is clear to us that combining task context and result ranking is a wholly unexplored area of CD research.

5. Summary

The IR field has been a named and identifiable research field for over 40 years. The two evaluation measures of recall and precision—used as early as 1958—have been widely applied in clone detection. However relatively little else has been applied, which leads one to wonder what has been missed. It is unwise to neglect this aspect because how one decides to evaluate a research product quite frequently drives at the heart of the research questions themselves.

Our position is that CD research has likely oversimplified its evaluation techniques. Now is a good time to investigate additional evaluation schemes. We believe some of the IR-inspired techniques are worthy of exploring in the context of CD research. But whether we adopt more IR-developed methods or not, we believe the issue of evaluation is critical, and that the field should not settle for continued use of simple precision and recall comparisons.

References

- [1] S. Bellon. Vergleich von Techniken zur Erkennung duplizierten Quellcodes. Master’s thesis, Fakultät Informatik, University of Stuttgart, Mar. 2002. Diplomarbeit Nr. 1998.
- [2] E. Burd and J. Bailey. Evaluating clone detection tools for use during preventative maintenance. In *Second IEEE International Workshop on Source Code Analysis and Manipulation (SCAM’02)*, pages 36–43, 2002.
- [3] K. Kontogiannis. Evaluation experiments on the detection of programming patterns using software metrics. In *Proceedings of the 1997 Working Conference on Reverse Engineering*, pages 44–54. IEEE Computer Society Press, 1997.
- [4] J. Mayrand, C. Leblanc, and E. M. Merlo. Experiment on the automatic detection of function clones in a software system using metrics. In *Proceedings of the IEEE Conference on Software Maintenance – 1996*, pages 244–254, 1996.
- [5] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 2nd edition, 1979.