BigCloneBench Considered Harmful for Machine Learning

Jens Krinke
UCL Computer Science
University College London
London, UK
0000-0003-1009-2861

Chaiyong Ragkhitwetsagul
SERU, Faculty of ICT
Mahidol University
Nakhon Pathom, Thailand
0000-0002-6502-1107

***Abstract*—BigCloneBench is a well-known large-scale dataset of clones mainly targeted at the evaluation of recall of clone detection tools. It has been beneficial for research on clone detection and evaluating the performance of clone detection tools, for which it has become standard. It has also been used in machine learning approaches to clone detection or code similarity detection. However, the way BigCloneBench has been constructed makes it problematic to use as ground truth for learning code similarity. This paper highlights the features of BigCloneBench that affect the ground truth quality and discusses common misperceptions about the benchmark. For example, extending or replacing the ground truth without understanding the properties of BigCloneBench often leads to wrong assumptions which can lead to invalid results. Also, a manual investigation of a sample of Weak-Type-3/Type-4 clone pairs revealed 86% of pairs to be false positives, threatening the results of machine learning approaches using BigCloneBench. We call for a halt in using BigCloneBench as the ground truth for learning code similarity.**

***Index Terms*—clone detection, code similarity, machine learning**

I. Introduction

The performance evaluation of clone detection or code similarity detection approaches is a common problem that needs well-constructed benchmarks. There have been a series of benchmarks [1]–[3] that have been used in the field of clone detection, however, most of them are of a limited scale due to the effort required in manually constructing the benchmarks. BigCloneBench [3]–[6] is a large-scale dataset of clones mainly targeted at the evaluation of recall of clone detection tools. Many papers have used BigCloneBench to evaluate and compare the performance of clone detection tools and it has become standard in the field of clone detection to evaluate the recall with BigCloneBench together with a manual evaluation of a sample to evaluate precision.

Recent approaches to clone detection and code similarity detection are based on machine learning from large-scale datasets. It is tempting to use BigCloneBench as the ground truth for learning code similarity. However, the way BigCloneBench has been constructed makes it problematic as the ground truth for such learning tasks. BigCloneBench has been created in a semi-automatic way with multiple steps. The semi-automatic approach leads to a situation in which the majority of the true positives have not been manually validated that they are indeed clones of each other. Moreover, only a small set of true negatives has been created and, for most of the possible pairs in the dataset, the ground truth is unknown.

The small size of the set of true negatives is a problem for the machine learning approaches because the ground truth is not representative and the way the true positives and negatives are constructed makes the ground truth biased. Not taking the imbalance and bias into account can lead to misleading results. Moreover, the way the dataset is constructed can lead to the misperception that any pair that is not in the ground truth of true positives is a true negative. This misperception can lead to invalid results and is a common problem in published papers.

The contributions of this paper are:

- A discussion of the features of BigCloneBench that affect the ground truth quality and a discussion of common misperceptions about the ground truth.
- A manual investigation of 100 random samples of the automatically constructed clone pairs.
- A discussion of the usage of BigCloneBench as ground truth for learning code similarity and how the issues and misperceptions affect the validity of results.

It is hoped that this paper will help the clone detection and code similarity communities to prevent producing invalid results by using BigCloneBench for machine learning.

II. BigCloneBench

BigCloneBench is available in multiple versions. An initial version only contained clones derived from 10 functionalities [4]. It was later expanded to include 43 functionalities [5], [7] in a framework to evaluate clone detection tools. The most detailed explanation is available in Svajlenko [6].

The discussion in this paper mainly uses the reported numbers [6] and will sometimes use numbers extracted from the released version of BigCloneBench which has been released individually[1] and as a part of the evaluation framework[2].

To understand the features of BigCloneBench that affect the ground truth quality, it is necessary to understand how BigCloneBench has been constructed. At the core of the

The authors would like to thank the BigCloneBench authors, Jeffrey Svajlenko and Chanchal Roy, for feedback and confirmation.

[1]https://github.com/clonebench/BigCloneBench
[2]https://github.com/jeffsvajlenko/BigCloneEval

To be published in the 16th International Workshop on Software Clones (IWSC '22), Limassol, Cyprus.

construction is the validation of methods by human judges: *Instead of asking the judges to validate if two code fragments are similar, they have been asked to validate if an individual function implements a target functionality. Clones are identified as code fragments that share functionality* [6]. The authors of BigCloneBench chose this approach to reduce subjectivity in judging whether two code fragments are clones of each other. We present the construction process of BigCloneBench in a simplified way in the following.

Source Code: BigCloneBench is built from the IJaDataset 2.0, a dataset of 250M LOC in 2.5M Java files from 25K projects mined from SourceForge and Google Code [6].

Exemplar Functions: The core of BigCloneBench is a set of 101 exemplar functions which have been composed as example implementations of 43 selected functionalities that are expected to appear often. 22 of the 43 selected functionalities have a single exemplar function. In earlier papers about BigCloneBench, these exemplar functions were termed "sample snippets", however, in this paper, we use the clearer term "exemplar functions" as it has been introduced in newer publications on BigCloneBench [3].

An example of a selected functionality is "Copy File" with the specification "copies a file". A corresponding exemplar function is shown in Figure 1. For the "Copy File" functionality, six exemplar functions have been created.

Potential Clones: From the exemplar functions and the specification of the 43 functionalities, a heuristic search has been performed to find methods that are using the 43 functionalities. The heuristic search has resulted in 77,933 methods.

For example, Figure 2 shows the search terms to identify methods that contain the "Copy File" functionality. Note that the search is not syntactically correct as given. 37,102 methods matched the search heuristic for the "Copy File" functionality.

Labeling of Potential Clones: A set of judges then compared the methods to the 43 functionalities and labelled them as true positives or false positives. Only a small number of methods (9,533 – 12%) have been labelled by more than one judge and the final label is the majority vote (887 methods with an equal number of votes are labelled as undecided).

The search heuristics have been designed so that they identify as many true positive snippets as possible without overburdening the judges in their tagging efforts. For the "Copy File" functionality, the heuristic search has identified 37,102 true positive snippets and a judge has labelled 3,084 of them as true positives and 34,018 as false positives. All snippets have been labelled by a single judge.

Overall, 15,290 methods have been labelled as true positives and 61,756 methods have been labelled as false positives. However, the heuristic search can retrieve the same method for more than one functionality. For example, snippet 22442270 has been labelled as a false positive for the "Copy File" functionality and as a true positive for the "Download From Web" functionality. Snippet 10151252 has been labelled as a true positive for the "Copy File" functionality because the snippet copies (uploads) a file to an FTP server and also as a true positive for the "Connect to FTP Server" functionality.

The released dataset has slightly different numbers and contains 75,673 retrieved methods. Of the retrieved methods, 73,906 are unique and 1,723 appear for more than one functionality. Moreover, from the 14,891 methods labelled as true positives, 14,679 are unique and from the 60,782 methods labelled as false positives, 60,019 are unique. 209 methods are true positives for more than one functionality.

Ground Truth: The ground truth[3] is constructed from the exemplar functions and the potential clones. The exemplar functions are in sets X_f for each of the 43 functionalities (f). The methods that are similar to the exemplar functions (true positives) are in sets P_f for each of the 43 functionalities (f) and the methods judged as false positives are in sets N_f.

The ground truth is constructed per functionality f, i.e., for pairs of methods $(m, n)_f$. The ground truth is constructed as follows: The pairs $(m, n)_f$ and $(n, m)_f$ are labelled as true positive if $(m \in X_f \cup P_f) \wedge (n \in X_f \cup P_f)$, i.e., if both methods are exemplar functions or labelled as true positives. The pairs $(m, n)_f$ and $(n, m)_f$ are labelled as false positive if an f exists such that $(m \in X_f) \wedge (n \in N_f)$. Note that false-positive pairs are only constructed for pairs of an exemplar function and a method labelled as true negative, i.e., pairs of two methods labelled as true negative are unlabelled pairs (as they could still be clones of each other).

For the running example of the "Copy File" functionality, 6 exemplar functions and 3,084 true-positive snippets lead to 4,772,505 pairs labelled as true clone pairs and 6 exemplar functions and 34,018 false-positive snippets lead to 204,108 pairs labelled as false clone pairs.

The above construction leads to a ground truth of 8,915,130 true clone pairs and 288,367 false clone pairs. None of the true or false clone pairs has been manually validated.

Automatic Classification: In the last step of the construction, the similarity of the methods in a clone pair is measured for every true clone pair. Based on the similarity, the clone pair is classified as Type-1 (T1), Type-2 (T2), Very-Strongly Type-3 (VST3), Strongly Type-3 (ST3), Moderately Type-3 (MT3), and Weakly Type-3/Type-4 (WT3/T4). It is worth noting that 8,498,894 out of 8,915,130 pairs have been classified as Weakly Type-3/Type-4 (95%).

Precision and Recall: The ground truth can be used to measure the recall of clone detection tools. BigCloneBench has not been intended for and should not be used directly to measure precision. The reason is that for most method pairs no ground truth is available and it is unknown whether the pair is a true positive or a false positive. Instead, precision needs to be evaluated differently, by manually investigating a representative sample of the results of the clone detection or code similarity detection. With the available ground truth data, only the lower and upper bound for the precision can be determined which can be used for a precision estimation.

[3]Technically, the BigCloneBench authors never use the term. However, as we focus on the use of BigCloneBench in machine learning where BigCloneBench is often used as the ground truth, we use the term ground truth for the labelled data in BigCloneBench.

```
public static void copyFile2(File srcFile, File destFile) throws IOException {
  FileUtils.copyFile(srcFile, destFile);
}
```

Fig. 1. Copy File Exemplar Function (Snippet 23677115 in file `CopyFileSamples.java`, lines 38–40).

```
[getChannel] OR [transferFrom]
OR [FileUtils.copyFile] OR [read AND write]
OR [nextLine AND [print OR println OR write]
    OR [IOUtils.copy] OR [IOUtils.copyLarge]
```

Fig. 2. Search terms for the "Copy File" functionality.

III. Observations

The construction of the ground truth leads to a few important observations which will be discussed in the following.

A. Relation Between Functionalities

Most importantly, the ground truth does not make any assumption on methods pairs where the methods appear in sets for different functionalities. For example, for a method pair (m, n) with $(m \in X_i \cup P_i) \wedge (n \in X_j \cup P_j) \wedge (i \neq j)$ it *cannot* be assumed that (m, n) is a true negative. This is highlighted by the fact that BigCloneBench allows methods to be in the true positive sets of different functionalities and indeed has 209 of such methods m with $(m \in P_i) \wedge (m \in P_j) \wedge (i \neq j)$.

(1) BigCloneBench does not contain information about pairs for different functionalities.

Any assumption that (m, n) is a true negative where the methods are from two different functionalities is not valid. Consider the two functionalities for "Copy File" and "Copy Directory". The functionality of copying a file is part of the functionality "Copy Directory" which copies a directory and its contents. At least 77 methods in the set of true positives for the functionality "Copy Directory" invoke a `copyFile` method. Therefore, all methods in the set of true positives for the functionality "Copy Directory" could be considered clones of methods for the "Copy File" functionality. Indeed, 19 methods in the set of true positives for the functionality "Copy Directory" have also been judged to be true positives for the "Copy File" functionality, and only three methods in the set of true positives for the functionality "Copy Directory" have been judged to be false positives for the "Copy File" functionality. However, it appears that the heuristic for the functionality "Copy File" has not retrieved most of the methods that have been retrieved for the functionality "Copy Directory".

(2) BigCloneBench does contain true but unlabelled clone pairs for different functionalities.

B. Relation Within Functionalities

Of lesser importance is the observation that the ground truth is even incomplete within the same functionality. As discussed above, the ground truth does not make any assumption on method pairs where both methods have been labelled as false positives for the given functionality. The reason is that such methods are not clones for any exemplar function of the given functionality, but they could be clones of each other.

(3) BigCloneBench does contain unlabelled true and false clone pairs for the same functionalities.

C. Ground Truth Quality

It has been reported [3], [6] that at least one judge disagreed with the others for 14.5% of the 9,533 methods that have been labelled by more than one judge. They extrapolate the average disagreement to estimate that at least 15% of the clones across the benchmark are subjective or have validation errors. Manual validation of clone pairs is also subjective [8].

An example of a wrongly labelled snippet for the "Copy file" functionality is snippet 19962035 (method `makeWF_BasicJavaWriterFormat_jwf`[4]) in file `184404.java`, lines 42–44. The snippet stores a constant string in a hash table (the text is source code that includes file operations) and does not copy a file.

There are many three-line methods for the copy file functionality. For some of them, there is only a weak semantic similarity. Figure 3 shows such a method. Its purpose is to dump the configuration to the standard output and the snippets in Figure 1 and Figure 3 would not be considered clones of each other as they are not textually similar (Types 1–3) or implementing the same functionality (Type 4).

Moreover, we identified at least 330 snippets in the "Copy File" functionality that do not contain the word 'file'. This is an indication that the snippets are not copying files but instead, for example, use `IOUtils.copy` to copy data between streams that are not files.

(4) BigCloneBench's labelled snippet ground truth quality is limited.

Another important observation is that the ground truth for method pairs (m, n) within the same P_f assumes that the two methods are indeed clones of each other or are similar to each other. No manual evaluation of this assumption has been done and therefore the quality of the ground truth for such pairs is unknown. Thus, at least some of the pairs (m, n) within the same P_f should not be considered true positives. This mostly affects the Weakly Type-3/Type-4 clone pairs since they account for 95% of all clone pairs in the ground truth.

Consider the functionality "Copy File" again which has six exemplar functions, most of them are very small with the smallest having only three lines of code[5] and the largest having 19 lines of code. The description of the functionality is simply "Copies a file". The smallest exemplar function for this functionality is shown in Figure 1.

[4]The method is too large to be shown here.

[5]Most of the code clone detectors usually detect clones in code snippets which are larger or equal to 6 lines.

```
private void dumpConfig() throws Exception {
  IOUtils.copy(new FileInputStream(m_snmpConfigFile), System.out);
}
```

Fig. 3. Method labelled as copy file functionality (Snippet 2571845 in file `1362837.java`, lines 51–53).

The largest method[6] judged to be a true positive for the copy file functionality is 762 lines long. This method does indeed contain functionality to copy a file, however, this is only a small part of the functionality. One can argue that the 762 LOC method and the 3 LOC method are somewhat related as both contain the functionality to copy a file. However, like the 762 LOC method, many methods in the set of true positives will not only have the functionality of copying a file but will also contain other functionalities. One should therefore not assume that methods judged to be true positives are necessarily clones of each other. This shows an issue in the manual tagging of the potential clones. Based on the way the dataset is constructed, an exemplar function is supposed to contain code that performs one specific functionality. By considering large potential clones, which can contain many functionalities, as true clones, the creation of true clone pairs is no longer valid in all cases.

(5) BigCloneBench's true clone pairs ground truth is flawed.

It is not clear to what extent the true clone pairs have been wrongly labelled as no manual checking of clone pairs has been done. Therefore we manually checked a random sample of 100 true clone pairs for the "Copy File" functionality classified as WT3/T4 clones. We checked whether the pair is indeed a true clone pair (according to the classification as Type-3/Type-4 clone [9], [10]) or not. We also checked whether the methods in the pair have been labelled correctly if we apply a stricter criterion: Does the method implement "Copies a file" as its main or only purpose? For example, methods that (A) encode, convert or rewrite, (B) read or write to a stream that is not a file, or (C) are (unit) tests are not considered to fulfil the requirement. The sample has been independently verified by both authors.

From the 100 samples[7], only 6 have been labelled as true positives by both authors, and 8 have been labelled differently by both authors, and 86 have been labelled as false positives by both authors. The number of false positives is surprisingly high and demonstrates a likely strong impact of the flawed ground truth construction of the ground truth quality. Although the sample is not representative and for a single functionality, the extremely high number of false positives indicates that the ground truth for WT3/T4 clone pairs is flawed. This is important for any evaluation of clone detectors in the WT3/T4 category, where the results can no longer be trusted. Even when we assume that only the WT3/T4 category of the "Copy File" functionality is affected, 52% of the ground truth (4,651,096 out of 8,915,130 snippets) should be discarded.

From the 192 methods in the 100 samples (8 appear twice), only 57 have been labelled as true positives by both authors, 120 of them have been labelled as false positives by both authors, and 15 of them have been labelled differently by both authors. Again, the number of false positives is surprisingly high which casts doubts on the ground truth quality created from the manual labelling of snippets.

D. Bias and Balance

As the 43 functionalities have hugely varying numbers of true and false positives, the ground truth is biased and imbalanced. For example, the majority of labelled methods in the released dataset are for the functionality "Copy File", 42,664 out of 75,672. Moreover, 5,935 out of 14,891 methods labelled as true positives are for the functionality "Copy File" (40%) leading to the situation that 4,664,949 out of 8,915,130 pairs considered to be true clone pairs are for the functionality "Copy File" (54%). Over 90% of all true clone pairs are just for eight functionalities and 22 out of the 43 functionalities taken together only amount to less than 1% of all true clone pairs. The imbalance is even worse for the false positive clone pairs where 70% are pairs for the functionality "Copy File". Over 98% are pairs just for eight functionalities. 32 out of the 43 functionalities taken together only amount to less than 1% of all false clone pairs.

(6) BigCloneBench is imbalanced and biased.

E. Impact

The six observations threaten the validity of evaluations done with the BigCloneBench benchmark or the BigCloneEval framework. They threaten the validity to different degrees. The incomplete ground truth (observations 1–3) and the bias and imbalance (observation 6) of the ground truth only have a limited impact on the evaluation of clone detectors. However, the bias and imbalance will have a strong impact on machine learning approaches for code similarity that learn from BigCloneBench's ground truth. In addition, as we will discuss in the following, some approaches have not considered that BigCloneBench does not contain information about pairs for different functionalities and wrongly assumed that pairs for different functionalities are not clones of each other.

As discussed above, the ground truth quality for labelled snippets is limited as at least 15% are estimated to be subjective or have validation errors. Moreover, we have discussed that the ground truth for true clone pairs is flawed leading to a strong impact on the validity of the ground truth for WT3/T4 clone pairs, threatening the validity of results for evaluations in the WT3/T4 category and approaches learning code similarity.

[6]Snippet 23094550, which is method `render` in file `402201.java`, lines 138–899.

[7]The samples and the results of the manual validation are available at https://github.com/jkrinke/BigCloneBench-Sample-Validation.

IV. Short Literature Study

To evaluate the impact of the observed BigCloneBench benchmark issues, we have performed a short literature study in which we checked published work for the following issues:

1) Has BigCloneBench been used for evaluation?
2) Is BigCloneBench used as ground truth for machine learning?
3) Has the ground truth been changed or extended?
4) Has BigCloneBench's ground truth been validated?

The first question is answered by extracting the papers that have actually used BigCloneBench for evaluation. The second question extracts papers that have used BigCloneBench as a dataset for machine learning. We only consider papers that use the true and false positive sets for machine learning and ignore papers that use the source code of BigCloneBench, but not the ground truth. For example, papers that learn code representations from the source code of BigCloneBench are not considered for this question [11]–[13]. The third question identifies whether the paper has changed the ground truth so that the oracle includes clone pairs from different functionalities. The last question checks whether the paper has validated the ground truth.

The third question is the most important one as it identifies papers that have used the ground truth in a way that is not consistent with the true positives and false positives provided by BigCloneBench. As discussed above, such ground truth use is very likely flawed.

Only a small set of published work has been investigated. We used IEEE Xplore to search[8] for papers that mention "`BigCloneBench`". This resulted in 75 papers. Of the 75 papers, 48 papers use BigCloneBench for evaluation purposes, and the other 27 papers present or discuss BigCloneBench (or just mention it as related work), but do not do any evaluation. Twenty papers use BigCloneBench as a dataset for machine learning where the machine learning uses the information of whether two fragments are cloned or not. However, at least five of the papers change or replace the ground truth. For example, by assuming that fragments of two different functionalities are not clones, or by assuming that all pairs that have not been labelled as true clones are not clones. Three of the five papers use a dataset [14] derived from BigCloneBench which has replaced the ground truth.

None of the papers discusses how BigCloneBench's ground truth construction affects the validity of the evaluations or the machine learning and we found no paper that validated the ground truth of clone pairs. More importantly, none of the papers that use machine learning uses a manually validated dataset in addition to BigCloneBench to evaluate the performance of the presented clone detection approaches.

Because the identified issues can have a large impact on the results' validity of machine learning using BigCloneBench, the next section will discuss the impact in more detail with a few examples.

[8]The search was done 28 June 2022.

V. Machine Learning with BigCloneBench

In recent years there has been a push on using machine learning to predict if two methods are clones or similar [11]–[32]. The success of machine learning approaches is dependent on having a sufficiently large ground truth to learn from, which, in addition, is balanced and representative. It is impossible to manually create such a ground truth and therefore other datasets have been used instead. Examples for such datasets are POJ/OJClone [33] or past submissions to the Google Code Jam[9]. In such datasets, participants create small programs to solve a given task. It is usually assumed that the solutions of a given task are similar to each other and that the solutions to different tasks are different. This assumption has not been manually verified but it is known that the quality of such a ground truth is affected by the way programmers are approaching the task. For example, participants in the Google Code Jam often use a set of utility functions that they use (and submit) in their solutions for every task.

A. Relation Between Functionalities

A similar assumption is also used for the BigCloneBench dataset by different approaches to machine learning of code similarity. In three of the eight papers discussed in the previous section, instead of only using the provided ground truth, the approaches construct their own ground truth and dataset by only considering the exemplar functions and the potential clones labelled as true positives and considering all methods for different functionalities as false positives.

For example, Wang et al. [19] follows this approach. Their approach uses an augmented AST-based representation and graph neural networks to measure the similarity of code pairs. The experiments are done with the Google Code Jam dataset and an earlier version of the BigCloneBench dataset. From the BigCloneBench dataset, only 9,134 methods are used. The paper reports that their ground truth contains 336,498 true positives and 2,080,088 false positives although the original ground truth has 6,164,953 true positives and only 258,574 false positives [4].

Wei and Li [14] use an AST-based LSTM network to learn the similarity of methods in which simultaneously the representation and a hash function are optimised such that hash codes for clone pairs are close to each other, and those for none-clone pairs are far away. It is not clear if the experiments only use the provided ground truth for learning or if they construct their own ground truth and dataset. However, they use the same subset of 9,134 methods as the BigCloneBench dataset which has been used in Wang's study [19]. Moreover, the paper presents precision results for the BigCloneBench dataset without discussion of a manual investigation.

We have approached the authors of both papers for clarification on how the ground truth has been constructed. However, we received no response. Both papers use a small subset of 9,134 fragments of the more than 50,000 fragments after discarding fragments without any tagged true and false clone

[9]https://codingcompetitions.withgoogle.com/codejam

pairs. However, the subset could not be reproduced and it is unclear how it has been achieved.

Another paper [18] uses the same subset and provides it[10]. The repository does not contain an explanation of how the subset was created, however, it allowed us to investigate the snippets and true clone pairs and the false clone pairs used in the machine learning. The investigation revealed that the created dataset expanded the ground truth provided by BigCloneBench without considering the issues we observed. We could confirm the presence of wrongly generated false clone pairs caused by assuming (A) two methods labelled as false positive for the same functionality are not clones of each other, and (B) two methods labelled as true positive for two different functionalities are not clones of each other.

Consider the snippet 22442270 again, which is a true positive for the "Download From Web" functionality and a false positive for the "Copy File" functionality. This snippet appears often as part of false clone pairs in the expanded ground truth. It appears in false clone pairs together with other snippets from the "Download From Web" functionality labelled as false positives (assumption A) and together with snippets from all other functionalities, both labelled as true (assumption B) and false positives (false positives that appear are true positives for other functionalities).

Also consider the snippet 10151252 again, which is a true positive for the "Copy File" functionality and also a true positive for the "Connect to FTP Server" functionality. It appears in false clone pairs together with snippets from all other functionalities, both labelled as true and false positives. It also appears in false clone pairs together with snippets that have not been labelled as positive for any functionality, only as a negative for a functionality (most often the "Download From Web" functionality[11]).

The above discussion shows that machine learning approaches using BigCloneBench are at risk when attempting to create complete ground truths, in particular when attempting to label method pairs with methods from different functionalities. Moreover, papers using the dataset we investigated are likely threatened in their validity [18], [20], [34].

B. Relation Within Functionalities

The potential presence of unlabelled clone pairs, i.e., pairs that have not been labelled as true or false clone pairs but could be true clone pairs, is a problem for the machine learning approaches. However, we could not identify any paper that restricts training and testing to pairs within the same functionality. CCLearner [16] only used the provided ground truth of the "Copy File" functionality for training, but used the provided ground truth of all functionalities for testing. Because of the presence of unlabelled pairs, the precision was evaluated by a manual check of a statistically significant sample size of the results produced by CCLearner.

[10]https://github.com/yh1105/datasetforTBCCD (the dataset contains 9,133 fragments, not 9,134)

[11]Some of them have clearly been labelled incorrectly in the original BigCloneBench ground truth.

C. Ground Truth Quality

As we discussed, BigCloneBench's ground truth for WT3/T4 cannot be trusted. As most machine learning approaches focus on Type-4 clones, their results are strongly impacted by the low quality of the ground truth, both in training their models and in testing their models. Not only is the validity of the results threatened for machine learning approaches using the ground truth, but also for approaches that are compared to previous approaches based on the results achieved with BigCloneBench. For example, a study by Yu et al. [20] shows that the semantic clones (MT3, WT3/T4) in BigCloneBench usually contain the same identifier names. Their experiment using a Linear-Model to detect semantic clone pairs in BigCloneBench only based on identifier names provides a comparable result to the state-of-the-art ML-based techniques such as ASTNN [17], TBCCD [18], and FA [19]. They also report that the performance of the three tools, trained on the original BigCloneBench, on a revised BigCloneBench dataset after abstracting the identifier names drops significantly (F1-score decreases 16%–27%). If the WT3/T4 ground truth could be trusted, their results would show a threat to external validity that the high-performing models based on the BigCloneBench training and evaluation may not perform well in practice where semantic clones do not contain similar identifier names. However, as their results are based on flawed ground truth, it is not clear if their results would be the same for a corrected ground truth.

D. Bias and Balance

As 95% of all true clone pairs in the ground truth belong to the WT3/T4 category, this category deserves some special discussion. Detecting Type-4 clones is very challenging and it is no surprise that classic clone detectors have a low recall in this category (some papers don't even discuss the category in their evaluations). On the other hand, machine learning approaches for clone detection have specifically targeted that category to identify semantically similar code. Some papers have even only used the WT3/T4 clone pairs for their machine learning. Most machine learning papers report good recall rates for the detection of WT3/T4 clones. However, as we have observed and discussed, the quality of the ground truth in the WT3/T4 category is low and it is not clear what the validity of the results is. Even ignoring the quality of the ground truth, the bias and imbalance of the ground truth is a problem. If the machine learning would mainly learn if both fragments of a pair implement some copy file functionality, it could achieve a good recall of at least 55% (as 4,651,096 out of 8,498,894 true WT3/T4 clone pairs are for the "Copy File" functionality).

VI. Conclusions

We have discussed how BigCloneBench's ground truth construction affects the validity of evaluations using BigCloneBench. We have observed that the ground truth is not just imbalanced and biased, but that the true clone pair ground truth is flawed as it contains clone pairs that a human judge would not consider to be cloned. Our manual investigation of

a random sample of 100 WT3/T4 clone pairs showed 86% of constructed clone pairs to be false positives (only 6% were true positives). Despite the sample being too small to be representative and was only done for one functionality, the very large number of 86% false positives suggests that the ground truth for WT3/T4 clone pairs cannot be trusted. However, as the main aim of BigCloneBench is to evaluate clone detectors for Type-1, Type-2, or Type-3 clones, the flawed ground truth for WT3/T4 clones is probably not a significant problem that would invalidate the results of the evaluation.

The discussed issues are a larger threat to machine learning approaches where the ground truth is used to learn whether a pair of code fragments are clones or not. Because machine learning approaches focus on WT3/T4 clone pairs, the results of machine learning approaches using BigCloneBench are threatened in their validity and cannot be trusted.

Moreover, we observed that some machine learning approaches address a large number of unlabelled code pairs by changing or replacing the ground truth by making wrong assumptions about fragments for different functionalities. By doing so, the approaches create a large number of code pairs wrongly labelled as not being cloned.

We hope to raise awareness of the flaws in the ground truth and how BigCloneBench's ground truth construction affects the validity of evaluations. The benchmark is still useful for its original purpose, i.e., evaluating code clone detection, when taking its limitations into account. Nonetheless, we call for a stop to using BigCloneBench for machine learning because the ground truth quality is too low to produce trustworthy results.

REFERENCES

[1] S. Bellon, R. Koschke, G. Antoniol, J. Krinke, and E. Merlo, "Comparison and evaluation of clone detection tools," *IEEE Transactions on Software Engineering*, vol. 33, no. 9, 2007.

[2] C. Ragkhitwetsagul, J. Krinke, and D. Clark, "A comparison of code similarity analysers," *Empirical Software Engineering*, vol. 23, no. 4, 2018.

[3] J. Svajlenko and C. K. Roy, "BigCloneBench," in *Code Clone Analysis*. Springer Singapore, 2021.

[4] J. Svajlenko, J. F. Islam, I. Keivanloo, C. K. Roy, and M. M. Mia, "Towards a big data curated benchmark of inter-project code clones," in *Int. Conf. on Software Maintenance and Evolution (ICSME)*, 2014.

[5] J. Svajlenko and C. K. Roy, "Evaluating clone detection tools with BigCloneBench," in *Int. Conf. on Software Maintenance and Evolution (ICSME)*, 2015.

[6] J. Svajlenko, "Large-scale clone detection and benchmarking," Ph.D. dissertation, University of Saskatchewan, 2017.

[7] J. Svajlenko and C. K. Roy, "BigCloneEval: A clone detection tool evaluation framework with BigCloneBench," in *Int. Conf. on Software Maintenance and Evolution (ICSME)*, 2016.

[8] F. Farmahinifarahani, V. Saini, D. Yang, H. Sajnani, and C. V. Lopes, "On precision of code clone detection tools," in *Int. Conf. on Software Analysis, Evolution and Reengineering (SANER)*, 2019.

[9] S. Carter, R. Frank, and D. Tansley, "Clone detection in telecommunications software systems: A neural net approach," in *Int. Workshop on Application of Neural Networks to Telecommunications*, 1993.

[10] C. K. Roy, J. R. Cordy, and R. Koschke, "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach," *Science of Computer Programming*, vol. 74, no. 7, 2009.

[11] K. Yokoi, E. Choi, N. Yoshida, and K. Inoue, "Investigating vector-based detection of code clones using BigCloneBench," in *Asia-Pacific Software Engineering Conf. (APSEC)*, 2018.

[12] D. Yuan, S. Fang, T. Zhang, Z. Xu, and X. Luo, "Java code clone detection by exploiting semantic and syntax information from intermediate code-based graph," *IEEE Transactions on Reliability*, 2022.

[13] Y. Gao, Z. Wang, S. Liu, L. Yang, W. Sang, and Y. Cai, "TECCD: A tree embedding approach for code clone detection," in *Int. Conf. on Software Maintenance and Evolution (ICSME)*, 2019.

[14] H. Wei and M. Li, "Supervised deep features for software functional clone detection by exploiting lexical and syntactical information in source code," in *Int. Joint Conf. on Artificial Intelligence*, 2017.

[15] M. White, M. Tufano, C. Vendome, and D. Poshyvanyk, "Deep learning code fragments for code clone detection," in *Int. Conf. on Automated Software Engineering (ASE)*, 2016.

[16] L. Li, H. Feng, W. Zhuang, N. Meng, and B. Ryder, "CCLearner: A deep learning-based clone detection approach," in *Int. Conf. on Software Maintenance and Evolution (ICSME)*, 2017.

[17] J. Zhang, X. Wang, H. Zhang, H. Sun, K. Wang, and X. Liu, "A novel neural source code representation based on abstract syntax tree," in *Int. Conf. on Software Engineering (ICSE)*, 2019.

[18] H. Yu, W. Lam, L. Chen, G. Li, T. Xie, and Q. Wang, "Neural detection of semantic code clones via tree-based convolution," in *Int. Conf. on Program Comprehension (ICPC)*, 2019.

[19] W. Wang, G. Li, B. Ma, X. Xia, and Z. Jin, "Detecting code clones with graph neural network and flow-augmented abstract syntax tree," in *Int. Conf. on Software Analysis, Evolution and Reengineering (SANER)*, 2020.

[20] H. Yu, X. Hu, G. Li, Y. Li, Q. Wang, and T. Xie, "Assessing and improving an evaluation dataset for detecting semantic code clones via deep learning," *ACM Transactions on Software Engineering and Methodology*, 2022.

[21] D. Wang, Z. Jia, S. Li, Y. Yu, Y. Xiong, W. Dong, and X. Liao, "Bridging pre-trained models and downstream tasks for source code understanding," in *Int. Conf. on Software Engineering (ICSE)*, 2022.

[22] N. D. Q. Bui, Y. Yu, and L. Jiang, "InferCode: Self-supervised learning of code representations by predicting subtrees," in *Int. Conf. on Software Engineering (ICSE)*, 2021.

[23] S. Zhou, B. Shen, and H. Zhong, "Lancer: Your code tell me what you need," in *Int. Conf. on Automated Software Engineering (ASE)*, 2019.

[24] Y. Wu, D. Zou, S. Dou, S. Yang, W. Yang, F. Cheng, H. Liang, and H. Jin, "SCDetector: Software functional clone detection based on semantic tokens analysis," in *Int. Conf. on Automated Software Engineering (ASE)*, 2020.

[25] C. Guo, D. Huang, N. Dong, Q. Ye, J. Xu, Y. Fan, H. Yang, and Y. Xu, "Deep review sharing," in *Int. Conf. on Software Analysis, Evolution and Reengineering (SANER)*, 2019.

[26] M. Tufano, C. Watson, G. Bavota, M. Di Penta, M. White, and D. Poshyvanyk, "Deep learning similarities from different representations of source code," in *Int. Conf. on Mining Software Repositories (MSR)*, 2018.

[27] W. Amme, T. S. Heinze, and A. Schafer, "You look so different: Finding structural clones and subclones in Java source code," in *Int. Conf. on Software Maintenance and Evolution (ICSME)*, 2021.

[28] L. Buch and A. Andrzejak, "Learning-based recursive aggregation of abstract syntax trees for code clone detection," in *Int. Conf. on Software Analysis, Evolution and Reengineering (SANER)*, 2019.

[29] C. Tao, Z. Qi, H. Xing, and X. Xia, "C4: Contrastive cross-language code clone detection," in *Int. Conf. on Program Comprehension (ICPC)*, 2022.

[30] W. Hua, Y. Sui, Y. Wan, G. Liu, and G. Xu, "FCCA: Hybrid code representation for functional clone detection using attention networks," *IEEE Transactions on Reliability*, vol. 70, no. 1, 2021.

[31] Y. Fujiwara, N. Yoshida, E. Choi, and K. Inoue, "Code-to-code search based on deep neural network and code mutation," in *Int. Workshop on Software Clones (IWSC)*, 2019.

[32] G. Mostaeen, J. Svajlenko, B. Roy, C. K. Roy, and K. A. Schneider, "On the use of machine learning techniques towards the design of cloud based automatic code clone validation tools," in *Int. Working Conf. on Source Code Analysis and Manipulation (SCAM)*, 2018.

[33] L. Mou, G. Li, L. Zhang, T. Wang, and Z. Jin, "Convolutional neural networks over tree structures for programming language processing," in *Thirtieth AAAI Conf. on Artificial Intelligence*, 2016.

[34] L. Chen, W. Ye, and S. Zhang, "Capturing source code semantics via tree-based convolution over API-enhanced AST," in *Int. Conf. on Computing Frontiers*, 2019.