Quest for the Golden Approach: An Experimental Evaluation of Duplicate Crowdtesting Reports Detection

Yuekai Huang[1,3], Junjie Wang[1,2,3,*], Song Wang[4], Zhe Liu[1,3], Yuanzhe Hu[1,3], Qing Wang[1,2,3,*]
[1]Laboratory for Internet Software Technologies, [2]State Key Laboratory of Computer Sciences,
Institute of Software Chinese Academy of Sciences, Beijing, China;
[3]University of Chinese Academy of Sciences, Beijing, China; *Corresponding author
[4]Lassonde School of Engineering, York University, Canada;
huangyuekai18@mails.ucas.ac.cn,junjie@iscas.ac.cn,wq@iscas.ac.cn

ABSTRACT

Background: Given the invisibility and unpredictability of distributed crowdtesting processes, there is a large number of duplicate reports, and detecting these duplicate reports is an important task to help save testing effort. Although, many approaches have been proposed to automatically detect the duplicates, the comparison among them and the practical guidelines to adopt these approaches in crowdtesting remain vague.
Aims: We aim at conducting the first experimental evaluation of the commonly-used and state-of-the-art approaches for duplicate detection in crowdtesting reports, and exploring which is the golden approach.
Method: We begin with a systematic review of approaches for duplicate detection, and select ten state-of-the-art approaches for our experimental evaluation. We conduct duplicate detection with each approach on 414 crowdtesting projects with 59,289 reports collected from one of the largest crowdtesting platforms.
Results: Machine learning based approach, i.e., ML-REP, and deep learning based approach, i.e., DL-BiMPM, are the best two approaches for duplicate reports detection in crowdtesting, while the later one is more sensitive to the size of training data and more time-consuming for model training and prediction.
Conclusions: This paper provides new insights and guidelines to select appropriate duplicate detection techniques for duplicate crowdtesting reports detection.

CCS CONCEPTS

• **Software and its engineering** → *Software testing and debugging*.

KEYWORDS

Crowdtesting, duplicate detection, information retrieval, machine learning, deep learning

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ESEM '20, October 8–9, 2020, Bari, Italy
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7580-1/20/10…$15.00
https://doi.org/10.1145/3382494.3410694

ACM Reference Format:
Yuekai Huang[1,3], Junjie Wang[1,2,3,*], Song Wang[4], Zhe Liu[1,3], Yuanzhe Hu[1,3], Qing Wang[1,2,3,*]. 2020. Quest for the Golden Approach: An Experimental Evaluation of Duplicate Crowdtesting Reports Detection. In *ESEM '20: ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (ESEM '20), October 8–9, 2020, Bari, Italy*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3382494.3410694

1 INTRODUCTION

Crowdtesting is an emerging paradigm which can improve the cost-effectiveness of software testing and accelerate its process, especially for mobile applications [1–3, 16, 27, 56]. It entrusts testing tasks to online crowdworkers whose diverse testing environments, background, and skill sets could significantly contribute to more reliable, cost-effective, and efficient testing results [2, 3]. Crowdtesting has been adopted by many software organizations, including but not limited to Google, Facebook, Amazon, Microsoft [4]. According to the latest statistics from Applause (also known as uTest)[1], benefits of leveraging crowdtesting include average increases on testing capacity by 200%, the number of releases per year by 150%, and average reduction of critical fixes by 50%.

In crowdtesting, crowdworkers are required to submit test reports after performing testing tasks. A typical report describes how the test was performed and what happened during the test. In order to attract workers, testing tasks are often financially compensated; thus workers may submit thousands of test reports due to financial incentive and other motivations [14, 15, 44, 45, 47, 50]. Given the invisibility and unpredictability of distributed crowdtesting processes, there are large number of duplicate reports. Previous study based on real industrial crowdtesting data revealed that an average of 82% crowdtesting reports are duplicates of others [46]. A significant problem with such a large number of duplicate reports is that the subsequent analysis by software testers becomes extremely complicated. For example, existing studies found that merely working through 500 crowdtesting reports to find the duplicate ones takes almost the whole working day of a tester [46].

Many approaches have been proposed to automatically detect the duplicate open source bug reports or duplicate sentences [9, 10, 12, 29–31, 36, 38–40, 53, 55, 57]. For example, [30, 36] applied information retrieval techniques for duplicate detection by computing the textual similarity between two reports. [39, 40, 57] designed a set of features for measuring the reports' similarity in terms of textual descriptions and attributes, and employed machine learning techniques for duplicate detection. [12, 29, 38, 53, 55] modeled

[1]https://www.applause.com/

the semantic similarity of reports using deep learning techniques for duplicate detection. Despite of that, different studies employed different mechanisms or features for duplicate detection, and evaluated on different datasets. Besides, none of them has conducted a complete comparison to explore which approaches are more effective, resulting in a lack of practical guides when using these previous approaches for duplicate crowdtesting reports detection.

This work aims at conducting an experimental evaluation of the commonly-used and state-of-the-art approaches for duplicate detection, and exploring which is the golden approach, i.e., the most effective, for crowdtesting reports. Specifically, we first conduct a literature review of duplicate detection approaches, and select ten state-of-the-art approaches as baselines. We conduct our experiments on 414 crowdtesting projects with 59,289 reports. Since obtaining training data is often time and effort consuming, we also investigate the sensitivity of these approaches to the training data size to evaluate these approaches under effort-aware scenarios. Besides, we also present the time cost for model training and prediction to better facilitate the selection of these approaches in real-world crowdtesting practice.

Results show that machine learning based approach, i.e., *ML-REP* [39], and deep learning based approach, i.e., *DL-BiMPM* [53], are the best two approaches of crowdtesting reports duplicate detection. Median $recall@1$ of these two approaches are 0.74 and 0.73 respectively, indicating in 74% and 73% cases they can find the duplicate reports with the first recommendation; and the $recall@5$ are 0.93 and 0.91 respectively, indicating in 93% and 91% cases they can find the duplicate reports with the first five recommendations. Furthermore, the best deep learning based approach, i.e., *DL-BiMPM*, is more sensitive to the size of training data and more time-consuming in model building and prediction than *ML-REP*. Among the deep learning based duplicate detection approaches, these adopt such network structure, i.e., modeling the interactive aspect of report pair, can achieve better performance than others.

This work provides new insights and guidelines for duplicate crowdtesting reports detection. Specifically, if there are few data or less time available for model training, we recommend *ML-REP* for duplicate detection; Otherwise, both *ML-REP* and *DL-BiMPM* are applicable. In addition, since deep learning is sweeping various fields, the more promising deep network structure found, i.e., modeling the interactive aspect, can serve as guidelines when designing new deep learning based approaches for duplicate detection.

This paper makes the following contributions:

- A rigorous evaluation of existing crowdtesting reports duplicate detection approaches. **To the best of our knowledge, this is the first work to extensively evaluate the duplicate detection approaches.**
- An extensive survey on approaches for duplicate detection of bug reports and general sentences.
- The comparison of ten state-of-the-art duplicate detection approaches on crowdtesting data, which can serve as the practical guidelines to choose approaches for duplicate crowdtesting reports detection.
- Public-access implementations[2] of the examined duplicate detection approaches to facilitate the replication of this study.

[2]https://doi.org/10.5281/zenodo.3852690

The rest of this paper are organized as follows. Section 2 and 3 respectively describe our literature review of duplicate detection and the selected ten duplicate detection approaches. Section 4 describes the crowdtesting dataset used in this work. Section 5 and 6 present the experimental design and obtained results. Section 7 discloses the threats to validity. Section 8 surveys related work. Finally, we summarize this paper in Section 9.

2 LITERATURE REVIEW FOR DUPLICATE DETECTION APPROACHES

To extensively survey existing applicable techniques for detecting duplicate crowdtesting reports, we conduct a Literature Review. We use Kitchenham's description [23] to extend our system review principles, which are described as follows.

2.1 Search Terms

We identify key terms used for the search from previous work [9, 10, 12, 29–31, 36, 38–40, 53, 55, 57] and our experience with the subject area. The search terms are as follows: *(report or bug or text or sentence) and (duplicate or duplication or similar or similarity or rank or match).* Note that, we noticed that there are duplicate sentences detection approaches which can also be utilized for duplicate crowdtesting reports detection, we set the search terms with *text or sentence* to include these studies for facilitate the thorough exploration of existing approaches.

2.2 Sources of Information

The following three databases are covered for retrieving related literatures: *DBLP, IEEE Xplore, and ACM digit library*, and the search is conducted in June 2019.

2.3 Study Selection

Selection of studies for inclusion in the literature review is a three-stage process: (1) initial selection of studies based on the title, (2) selection of studies after reading the abstract, and (3) further selection of studies after reading the paper.

Since our literature review aims at selecting the studies which can be used in the duplicate detection of crowdtesting reports, we introduce the following inclusion and exclusion criteria to facilitate the selection process. Studies selected at each stage of the selection process meet our inclusion criteria:

- Full peer-review papers with validation results.
- Propose a method of duplicate detection for bug reports or sentences.
- Have detailed method description.

Studies rejected at each stage of the selection process meet our exclusion criteria:

- Papers unrelated to duplicate detection.
- Theoretical research about duplicate detection.
- Refinements or enhancements to duplicate detect approaches.

In summary, we started with 41,196 records from database search and limit the search to the title, we get 2,911 records and enter the first phase. In the first phase, we selected papers based on the title, and 110 papers were selected and moved to the second phase for further selection. In the second phase, these studies are mainly

Table 1: Overview of ten duplicate detection approaches

Abbreviation	Title	Venue	Category	Reference
IR-TF	Detection of Duplicate Defect Reports Using Natural Language Processing	ICSE 2007	Information Retrieval	[36]
IR-DBTM	Duplicate Bug Report Detection with a Combination of Information Retrieval and Topic Modeling	ASE 2012	Information Retrieval	[30]
ML-ADMA	A Discriminative Model Approach for Accurate Duplicate Bug Report Retrieval	ICSE 2010	Machine Learning	[40]
ML-REP	Towards More Accurate Retrieval of Duplicate Bug Reports	ASE 2011	Machine Learning	[39]
ML-BugSim	Learning to Rank Duplicate Bug Reports	CIKM 2012	Machine Learning	[57]
DL-DCNN	Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks	SIGIR 2015	Deep Learning	[38]
DL-MALSTM	Siamese Recurrent Architectures for Learning Sentence Similarity	AAAI 2016	Deep Learning	[29]
DL-CWEIR	Combining Word Embedding with Information Retrieval to Recommend Similar Bug Reports	ISSRE 2016	Deep Learning	[55]
DL-BiMPM	Bilateral Multi-Perspective Matching for Natural Language Sentences	IJCAI 2017	Deep Learning	[53]
DL-CRNN	Towards Accurate Duplicate Bug Retrieval using Deep Learning Techniques	ICSME 2017	Deep Learning	[12]

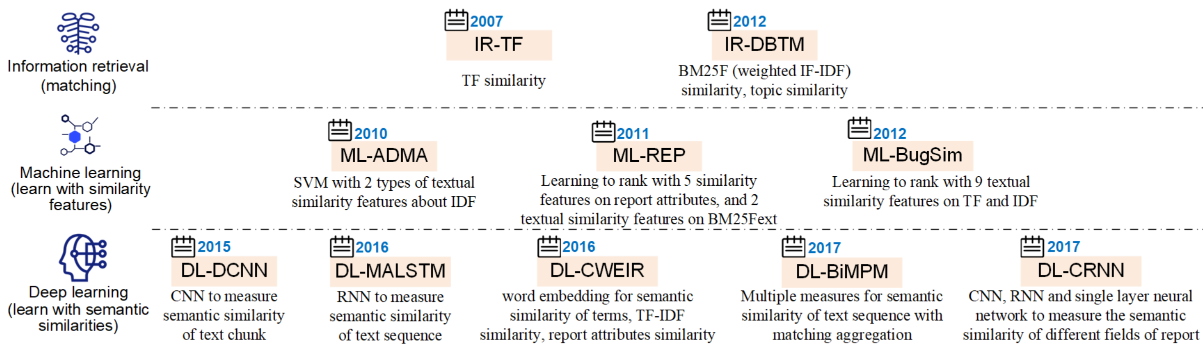

Figure 1: Summarized overview of the ten duplicate detection approaches

reviewed through their abstracts and 82 related papers are selected for further reading. Finally, 31 papers are selected and move to quality assessment.

2.4 Selection Verification

The first author did the selection of the studies according to the criteria outlined above. At each selection stage, the second author validated the selection of the studies. In the first phase, 300 of the studies were randomly selected for the second author to validate. 293 of the studies have the same decision by the first and second author. The discrepancy is caused by the differences between the interpretations of whether an approach is a refinement of duplicate detect method, such as [41]. However, we find that all these paper were not included in the final set, so it did not influence the results.

In the second stage, the second author evaluated the abstracts for 50 randomly selected studies. 47 of the selected studies have the same decision by the two authors. All the differences were between the studies the first author selected but the second author did not. It is worth mentioning that they are not included in the final set.

We also conducted similar selection verification process for the third stage, and all the selected studies has the same decision by the two authors.

2.5 Study Quality Assessment

For the final selected researches, we answer the following questions to assess their quality:

- Is there a clear stated research goal for duplicate detection?
- Is there a detailed method description?
- Is there a clear methodology for validating the approach?
- Are the subject projects selected for validation suitable (e.g., large enough for demonstrate the effectiveness of the approach) for the research goals?
- Is the selected evaluation metrics reasonable?
- Are there control techniques or baselines to demonstrate the effectiveness?
- Are the presented results clear and relevant to the research goals?
- Are the limitations of the approach enumerated?
- Is there any explicit contribution to duplicate detection?
- Can it be directly used in the crowdtesting reports?

All the above questions can be answered in three ways: *yes*, *no*, and *to somewhat*, corresponding to 1/0/0.5 points. The sum of the scores is used to measure the research quality of the selected literature. For the 31 selected papers[3] after the selection procedure, we first filter 15 papers with an quality score below 8.0. We then remove 2 papers that require specific information [18, 52], e.g., [18] employed contextual information which does not exist in crowdtesting environment. Besides, for the approaches of duplicate sentence detection, if there are a similar approach for duplicate reports detection, we also remove them, i.e., [20, 22, 42, 43]. Finally, 10 approaches are remained for further experimental evaluation, which will be described in detail in Section 3.

3 APPROACHES OVERVIEW AND PREPARATION

Table 1 and Figure 1 present an overview of the ten duplicate detection approaches. According to their utilized techniques, we group

[3]We list these papers in https://doi.org/10.5281/zenodo.3852690 to facilitate future studies.

them into three categories and rank them by their publication time in each category. Note that, for the abbreviation of these approaches, we prefix them with the abbreviation of category name for better understanding. Another note is that, for the approaches which do not have a name in their original paper, e.g., [12, 36, 38, 40, 55], we name them based on their main idea to facilitate reading.

3.1 Information Retrieval (IR) based Approaches

The first category contains two approaches, i.e., *IR-TF* [36] and *IR-DBTM* [30]. They apply information retrieval (IR) techniques for duplicate detection by computing the similarity between two reports.

IR-TF is the first work in duplicate bug reports detection. It computes the term frequency (TF) for each report and employ the cosine similarity to determine whether two reports are duplicate.

IR-DBTM improves *IR-TF* from two aspects. Firstly, it replaces term frequency with BM25F [33] which is an advanced document similarity function based on the weighted term frequency (TF) and inverse document frequency (IDF) to accurately measure the global and local importance of a term.

Secondly, it includes the topic similarity of two reports to address the problem of textual dissimilarity between duplicate reports. The Latent Dirichlet Allocation (LDA) [6] topic model is employed, and these two similarities are linearly combined with the weight learned with a searching algorithm by maximizing the *mAP* in training set.

3.2 Machine Learning (ML) based Approaches

The second category contains three approaches, i.e., *ML-ADMA* [40], *ML-REP* [39], and *ML-BugSim* [57]. They design a set of features for measuring the reports' similarity from various aspects, and employ machine learning algorithms to determine whether two reports are duplicate.

ML-ADMA designs 2 types of features to measure the textual similarity between two reports and builds machine learning model to automatically determine the duplicate status. It computes the inverse document frequency (IDF) for each term, finds the shared terms of two reports, and treats the sum of their IDF values as the feature (i.e., the first type of feature). The second type of feature is obtained in a similar way but in terms of bi-gram (e.g., treat two adjacent terms as one term). Different combinations of report's fields (e.g., summary, description) for these two feature types finally generate 54 features for the machine learner.

ML-REP improves *ML-ADMA* from two aspects. Firstly, it includes five attributes of reports (e.g., the *component* where the bug resides, the *priority* of the report), and designs five features to measure the similarity of these categorial information. Secondly, to better measure the textual similarity, it extends BM25F model (BM25Fext) [33] for structured long queries to better fit duplicate reports detection problem.

The BM25Fext scores are calculated based on uni-gram and bi-gram respectively, and treated as two features in the learning model. Seven features are imported into a learning to rank model to determine the duplicates among reports, in which RNC (a ranking cost function) [11] as loss function is optimized by gradient descent method. Note that, since crowdtesting reports do not have some of the five attributes, we use two counterparts, i.e., task id, priority of bug, for similarity measurement.

ML-BugSim argues that many of the report's attributes may be unknown at the time of submission, thus it only utilizes textual information for duplicate detection. It designs nine textual similarity features considering the term frequency (TF), inverse document frequency (IDF), and their combinations for both report's summary and description. A learning to rank model is utilized for predicting the duplicate score.

3.3 Deep Learning (DL) based Approaches

The third category contains five approaches, i.e., *DL-DCNN* [38], *DL-MALSTM* [29], *DL-CWEIR* [55], *DL-BiMPM* [53], and *DL-CRNN* [12]. The main idea of them is to model the semantic similarity of reports with deep learning techniques for duplicate detection.

DL-DCNN builds convolutional neural network (CNN) to measure the semantic similarity of text chunk in two reports. In detail, the textual descriptions are first encoded using CNN, and similarity is calculated between each pair of reports through a similarity layer. Meanwhile, additional features, i.e., IDF values and term overlaps, are extracted based on all terms and terms removing stopwords. Finally, the output of CNN layer, the similarity output by similarity layer, and additional features are jointly input into the softmax layer for determining the duplicate status. To ensure the correctness of our implementation, we refer to two open source implementations on GitHub[4][5].

DL-MALSTM argues the sequential information of the textual description should be considered, therefore it builds recurrent neural network (RNN) to measure the semantic similarity of text sequence in two reports. In detail, the textual descriptions are encoded using a siamese Long Shot Term Memory (LSTM) network (a special RNN structure) [8, 19], in which the text of two reports are input into the same LSTM network, and encoded into vectors, then the Manhattan distance between the two reports is calculated.

DL-CWEIR utilizes both information retrieval and deep learning techniques for duplicate detection. It combines three similarity scores to determine whether two reports are duplicates. The first similarity is based on the term frequency (TF) and inverse document frequency (IDF). The second similarity is obtained with word embedding technique which focuses more on the relationship of terms considering the context they appear. Following its original work, we use word2vec to pre-train word embedding model and convert the text to a word embedding vector for similarity calculation. The third similarity is based on two report's attributes, i.e., the *component* where the bug resides and the *product* of the report. We use *task id* (see Table 2), which is the most similar one, to substitute these two attributes.

DL-BiMPM also measures the semantic similarity of text sequence in two reports as *DL-MALSTM*. To improve the performance, it includes matching-aggregation [51, 54] to match the two text sequences from multiple perspectives. In detail, Bi-directional Long Shot Term Memory (Bi-LSTM) [19, 37] is used to encode the textual descriptions, then multi-perspective matching layer is utilized for encoding the output of each time step to the matching vectors.

[4] https://github.com/aseveryn/deep-qa
[5] https://github.com/gvishal/rank_text_cnn

The aggregation is then added to generate the matching vectors, and finally the softmax layer is used for determining the duplicate status. To ensure the credibility of this study, we re-use the author's implementation on GitHub[6] directly.

DL-CRNN employs different types of neural network to measure the semantic similarity of different fields of a report, i.e., CNN for encoding long descriptions, RNN for short descriptions, and single layer neural network for report's attributes (i.e., *task id* and *priority* in this study). After calculating the three types of encodings, a simple concatenation is made to form the final encoding, and then the cosine similarity between two final encoding will be calculated.

Before implementing these approaches for experimental evaluation, as previous approaches are designed and experimented on English dataset while our crowdtesting dataset is in Chinese, we additionally conduct the following data preprocessing steps. First, we use the Jieba Chinese word segmentation tool[7] to tokenize the summary and description into terms. Secondly, we remove the stopwords, e.g., the and on, to reduce noise. After that, the summary and description of crowdtesting reports are expressed as a list of terms. For the approaches which do not distinguish the fields of textual descriptions, we concatenate the summary with description to generate the overall textual description of a crowdtesting report.

4 SUBJECT PROJECTS

We have collected 414 crowdtesting projects with 59,289 crowdtesting reports from one of the largest crowdtesting platforms[8], which were closed between January 2015 and August 2016.

Figure 2 shows the statistics of these projects. There are a median of 108 crowdtesting reports in a project, and one report has a median of 7 duplicates. Each report is described with a median of 33 terms.

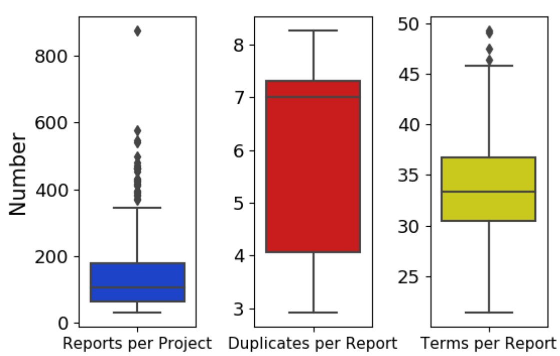

Figure 2: Detailed statistics of the dataset

Table 2 presents an example of crowdtesting reports. It contains a report ID, a task ID (i.e., which task is conducted), a crowd worker ID (i.e., who submit the report), the bug summary and reproduce steps of how the test was performed and what happened during the test, as well as the submission time. It also includes a duplicate tag assigned by the test engineer to indicate with which the report is duplicate. Note that, a typical bug report in open source software contains two textual fields, i.e., summary and description, which are commonly-used by existing duplicate detection approaches. To facilitate reading, we call the reproduce steps of a crowdtesting report as the description in the following paper.

Table 2: An example of crowdtesting report

Field Name	Example Value
Report ID	R1002987362
Task ID	T00008
Crowd worker id	W1000917120
Summary	Stop when downloading about 90%
Reproduce steps (Description)	Follow the steps to download. When it reaches 90%, the progress stops. Click pause/start repeatedly, but the download does not continue
Priority	P2
Submission time	2015/6/16 20:28:15
Duplicate tag	4

As three of the examined approaches, i.e., *DL-MALSTM*, *DL-DCNN*, and *DL-BiMPM*, are originally designed for the duplicate sentences detection, in order to provide a more thorough view of the performance for the ten examined approaches, we also experiment with the dataset of duplicate sentences. In detail, we employ the SNLI [7] dataset with 570k human written English sentence pairs. It is commonly-used in sentence matching tasks [25, 26] and has been utilized for evaluating our examined *DL-BiMPM* [53].

5 EXPERIMENT DESIGN

5.1 Research Questions

This paper aims at answering the following three research questions:

- **RQ1: Which approach is more effective for crowdtesting reports duplicate detection?**

As we have shortlisted ten applicable approaches for duplicate detection in Section 3, in this question, we examine the performance of these ten duplicate detection approaches on detecting duplicate crowdtesting reports. We also experiment on duplicate sentences to obtain a more comprehensive understanding of these approaches.

- **RQ2: Are these approaches sensitive to the size of the training data?**

Obtaining training data for duplicate crowdtesting report detection is often time and effort consuming, we also investigate the sensitivity of these approaches to the training data size to evaluate these approaches under effort-aware scenarios.

- **RQ3: What is the time cost of these duplicate detection approaches?**

Building machine learning or deep learning based solutions is often reported time consuming, which could limit their generalizability in real-world applications [28]. As most of the ten selected state-of-the-art duplicate detection approaches are based on either machine learning or deep learning techniques, this research question explores the time cost of model building and duplicate detection of the studied approaches to present a more comprehensive view of these approaches.

[6] https://github.com/zhiguowang/BiMPM
[7] https://github.com/fxsjy/jieba
[8] Baidu CrowdTest crowdtesting platform

5.2 Experiment Setup

To answer RQ1, for the 414 experimental crowdtesting projects (see details in Section 4), we randomly select 300 projects and use them for training, while use the left 114 projects for testing. We run each duplicate detection approach on each project from the testing dataset to conduct performance comparison. We also experiment with duplicate sentences dataset SNLI (see Section 4), and use the original separation of training sets and test sets provided in the dataset. Most of the examined duplicate detection approaches have many parameters that could affect their performance. In this work, for each approach, we adopted hyperparameter tuning to find the best values of each parameter involved. To answer RQ2, we experiment with different number of projects as training dataset while keep using the 114 projects as testing dataset (same as RQ1). Specifically, we respectively use the first 20, 50, 100, 150, and 300 (same as RQ1) projects as training dataset to investigate the performance variation for each duplicate detection approach. All the above experiments are conducted on a personal computer with CPU Intel(R) Core(TM) i9 3.1 GHz PC with 32GB RAM running Windows10 OS (64-bit). We record the time to run these experiments for answering RQ3.

Among the ten duplicate detection approaches, *DL-MALSTM*, *DL-DCNN*, *DL-BiMPM* and *ML-ADMA* require a set of positive instances and negative instances as training data. We treat each pair of reports with the same duplicate tag (see Table 2) as the positive instances. To keep the data balance, we randomly sample equal number of instances with two reports of different duplicate tags and treat them as negative instances. For approaches with learning to rank algorithm (i.e., *ML-REP* and *ML-BugSim*) and approaches using BM25F (i.e., *IR-DBTM* and *ML-REP*), the training data should be organized in the form of triplets, i.e., (report r_i, duplicate report of r_i, non-duplicate report of r_i). For each report, we treat the reports with the same duplicate tag as duplicate report, and random choose the report with different duplicate tags as non-duplicate report to build the training data.

When conducting duplicate detection on the testing dataset, for deep learning based approaches, *ML-ADMA* and *IR-TF*, we input all pairs of reports in a crowdtesting project to the model, obtain the duplicate probability or similarity of each pair, and use the value for determining the duplicates. For the approaches with learning to rank algorithm (i.e., *ML-REP* and *ML-BugSim*) and approaches using BM25F (i.e., *IR-DBTM* and *ML-REP*), we obtain the duplicate score of each pair, and use the score to determine the duplicates.

5.3 Evaluation Metrics

Following existing studies [12, 30, 36, 39, 40, 55, 57], we use *recall@k* and *mAP* as the metrics for measuring the performance of duplicate detection approaches.

Given a query report q, we set its ground truth duplicate reports as $G(q)$, and the top-k reports recommended by a duplicate detection method as $R(q)$. *Recall@k* checks whether there exist duplicate reports in top-k recommendation. We define *Recall@k* as follows:

$$recall@k = \begin{cases} 1, \ if \ G(q) \cap R(q) \neq \emptyset \\ 0, \ Otherwise \end{cases} \tag{1}$$

According to the formula, if there exists at least one ground truth duplicate report in the top-k recommendation, the top-k recommendation is useful for the query report q. Given a set of query reports in a crowdtesting project, we use the average *recall@k* of all query reports to measure the performance. Following existing studies [30, 39, 40, 55, 57], we experiment k with values of 1, 3, 5, and 10 to obtain the corresponding performance.

mAP (Mean Average Precision) is defined as the mean of the Average Precision (*AP*) values obtained for all the evaluation queries. The *AP* of a single query report q is calculated as follows:

$$AP(q) = \sum_{n=1}^{|G(q)|} \frac{Precision@k(q)}{|G(q)|} \tag{2}$$

In the above formula, $Precision@k(q)$ is the predicted precision over all the top-k reports in the ranked list, i.e., the ratio of ground truth duplicate reports of the query report q in the top-k recommendations. We calculate $Precision@k(q)$ with the following equation, where n is an iterator of all ground truth duplicates reports $G(q)$ in Equation 2 and 3.

$$Precision@k(q) = \frac{\#ground\ truth\ in\ top\ k}{n} \tag{3}$$

6 RESULT ANALYSIS

6.1 Answering RQ1: Performance Comparison

6.1.1 ***Performance Comparison on Crowdtesting Data.*** Figure 3 shows the performance of the ten investigated duplicate detection approaches on 114 crowdtesting projects. We also conduct Mann-Whitney Test between each pair of these approaches, and rank them based on their performance and whether the performance difference is significant ($p - value < 0.05$), with results in Table 3. Specifically, we start from the approach with the highest performance, and put these approaches with which no significant performance is observed ($p - value > 0.05$) into the same ranking bucket. We then iterate the process on the remaining approaches. *Ri* in Table 3 denotes the approaches in the i_{th} ranking.

Table 3: Performance ranking of the ten approaches (RQ1)

Ranking	Recall@1	Recall@3	Recall@5	Recall@10	mAP
R1	DL-BiMPM ML-REP DL-CWEIR	ML-REP DL-BiMPM DL-CWEIR	ML-REP DL-BiMPM	ML-REP	DL-BiMPM ML-REP
R2	ML-BugSim IR-DBTM	IR-DBTM ML-BugSim	IR-DBTM ML-BugSim DL-CWEIR	DL-BiMPM IR-DBTM ML-BugSim IR-TF DL-DCNN DL-CWEIR	ML-BugSim DL-CWEIR IR-DBTM DL-DCNN
R3	IR-TF DL-DCNN	IR-TF DL-DCNN	IR-TF DL-DCNN	ML-ADMA	IR-TF ML-ADMA
R4	ML-ADMA	ML-ADMA	ML-ADMA	DL-MALSTM	DL-CRNN DL-MALSTM
R5	DL-CRNN	DL-CRNN DL-MALSTM	DL-CRNN DL-MALSTM	DL-CRNN	–
R6	DL-MALSTM	–	–	–	–

ML-REP is the only approach which is ranked at the first level in all evaluation metrics, indicating it is the best duplicate detection approach on our crowdtesting dataset. The median *recall@1* is 0.74, indicating in 74% cases, *ML-REP* can find the duplicate reports

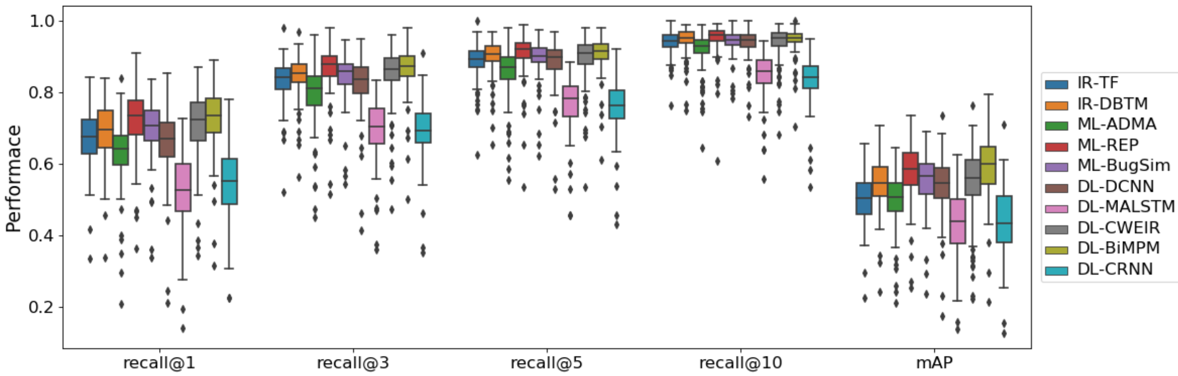

Figure 3: Performance of the ten duplicate detection approaches (RQ1)

with the first recommendation. The median *recall*@5 is 0.93, indicating in 93% cases, it can find the duplicate reports with the first five recommendations. This is promising in reducing the effort of project managers for manually examining the duplicate reports. *ML-REP* extracts similarity features and utilizes machine learning technique for duplicate detection. There might be two reasons for its superiority in duplicate detection performance. Firstly, it utilizes BM25F model which is an advanced document similarity function compared with the original term frequency and inverse document frequency, and it extends the model for structured long queries to better fit duplicate reports detection problem. Secondly, it includes features to measure the similarity of reports' attributes which can further help overcome the problem of textual dissimilarity between duplicate reports.

DL-BiMPM is the second best duplicate detection approach which is ranked at the first level in four metrics, and second level in the other metric. It models the semantic similarity of reports with deep learning techniques for duplicate detection. Compared with other approaches with deep learning techniques, it has better encoding and similarity calculation treatment. In detail, it uses Bi-LSTM [19, 37] to encode the text which can model the text sequential information compared with *DL-DCNN*. After encoding, it employs match-aggregation framework to calculate the similarity which can get the interactive features between texts from multiple perspectives compared with *DL-MALSTM* or *DL-CRNN*.

Other approaches with promising results are *DL-CWEIR*, *IR-DBTM*, *ML-BugSim* which are ranked in the first and second level in most evaluation metrics. Although these three approaches belong to different categories, what they have in common is they utilize advanced term matching strategies for measuring the textual similarity of reports. In detail, *DL-CWEIR* utilizes word embedding technique to capture the relationships of terms considering the context they appear. *IR-DBTM* not only utilizes BM25F for advanced similarity measurement but also incorporates topic model to further capture the similarity beyond text matching. *ML-BugSim* designs nine features to capture different aspects of textual similarity.

We also notice that two of the deep learning based approaches, i.e., *DL-MALSTM* and *DL-CRNN*, are ranked lower in all evaluation metrics. We will further analyze possible reasons in Section 6.1.3.

Further exploration in terms of projects. From Figure 3, we also observe that the duplicate detection performance varies among crowdtesting projects. We examine the projects with low performance, and find that in these projects, duplicate reports are expressed with different terms, while non-duplicate reports share a large portion of terms, which makes it hard to detect the duplicate reports accurately. This can be influenced by many factors, e.g., the number of functions under test, the number of paths of the application, etc. For example, if there exist multiple paths in a function, the reproduce steps for duplicate bugs related to this function can be described diversely which would confuse the duplicate detection approaches.

6.1.2 ***Performance Comparison on Duplicate Sentences Dataset***.

To better examine the performance of these duplicate detection approaches, we also run them in duplicate sentences dataset (see details in Section 4), with the results in Table 4. To calculate our applied metrics (*Recall@k* and *mAP*), we need to collect all the duplicates of each sentence which is not provided in our applied duplicate sentences dataset.

Instead, we use precision, recall, and F1 score to measure the performance of these approaches on the duplicate sentences dataset used in existing studies [29, 53].

Table 4: Performance of the ten selected duplicate detection approaches on the general sentence dataset.

Ranking by F1	Method	Precision	Recall	F1 Score
1	DL-BiMPM	90.60%	88.24%	89.41%
2	DL-DCNN	81.33%	76.58%	78.88%
3	DL-CRNN	74.78%	69.26%	71.91%
4	DL-MALSTM	76.15%	66.36%	70.92%
5	ML-BugSim	62.91%	80.26%	70.54%
6	ML-REP	63.26%	77.60%	69.70%
7	IR-DBTM	51.31%	69.85%	59.17%
8	DL-CWEIR	41.99%	87.20%	56.69%
9	ML-ADMA	43.66%	64.35%	52.02%
10	IR-TF	51.90%	48.65%	50.22%

The top four ranked approaches on duplicate sentences dataset are all deep learning based approaches, i.e., *DL-BiMPM*, *DL-DCNN*, *DL-MALSTM* and *DL-CRNN*. It is quite different from the performance on crowdtesting dataset, in which three of them (i.e., *DL-DCNN*, *DL-MALSTM* and *DL-CRNN*) is ranked almost in the rear of all approaches in Table 3. This indicates that the deep learning based approaches are the most promising ones for detecting duplicate sentences, while this is not the case for crowdtesting reports. It might because the local bias issue of crowdtesting projects which have been noticed in crowdtesting reports classification task [44, 47]. Specifically, reports from different domains of testing applications would use quite different terms, for example, *navigation*, *location* in map domain, and *lyric*, *song* in music domain. When conducting duplicate detection in testing projects with emerging terms in new domains, the above mentioned approaches might fail to take effect.

6.1.3 *Performance Difference of Deep Learning based Approaches*.

As deep learning is sweeping various fields, we further examine the deep learning based approaches to seek for guidelines when designing new deep learning based approaches for duplicate detection. For the four deep learning approaches which have well-designed network structures (except *DL-CWEIR*), we find that, no matter in crowdtesting dataset and general sentences dataset, *DL-MALSTM* and *DL-CRNN* are worse than *DL-BiMPM* and *DL-DCNN*. We further analyze the possible reasons and find that the basic network structure of the first two approaches is different from that of the last two approaches.

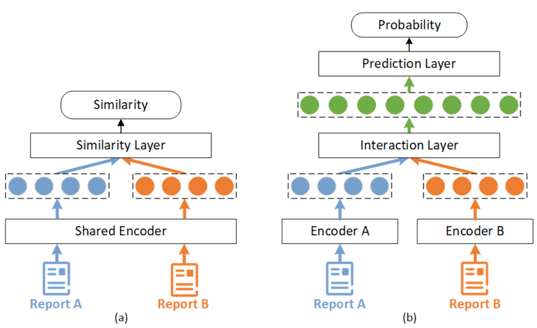

Figure 4: Two types of network structure of the deep learning based approaches.

Specifically, *DL-BiMPM* and *DL-DCNN* would first encode the single report, then use the interactive representation of the two encoded reports for further feature extraction and similarity prediction, as shown in Figure 4 (b). On the contrary, *DL-MALSTM* and *DL-CRNN* only encode the single report for similarity computation, as shown in Figure 4 (a). Hence, without utilizing the interactive information of pair of reports, *DL-MALSTM* and *DL-CRNN* lose important sources of information to accurately detect the duplicates.

1) Machine learning based approach, i.e., *ML-REP*, and deep learning based approach, i.e., *DL-BiMPM*, are the two best duplicate detection approaches. 2) Deep learning based approaches which perform good in duplicate sentences detection might fail in duplicate crowdtesting reports detection due to local bias issues. 3) Among the deep learning based approaches, these with such network structure, i.e., modeling the interactive aspect of report pair, can achieve better performance than others.

6.2 Answering RQ2: Sensitivity to Training Size

Figure 5 presents the performance of the ten duplicate detection approaches with different training sizes. Table 5 shows the ranking of their performance based on the results of Mann-Whitney Test as RQ1. Due to space limit, we only present the results of *mAP*, and other metrics demonstrate a similar trend.

Table 5: Ranking of the ten approaches with different training sizes based on mAP (RQ2)

Ranking	TS = 20	TS = 50	TS = 100	TS = 150	TS = 300
R1	ML-REP DL-CWEIR	ML-REP	ML-REP	DL-BiMPM ML-REP	DL-BiMPM ML-REP
R2	IR-DBTM	IR-DBTM DL-CWEIR	DL-BiMPM IR-DBTM DL-CWEIR ML-BugSim	ML-BugSim DL-CWEIR IR-DBTM	ML-BugSim DL-CWEIR IR-DBTM DL-DCNN
R3	ML-BugSim IR-TF	ML-BugSim	DL-DCNN IR-TF	DL-DCNN	IR-TF ML-ADMA
R4	ML-ADMA	IR-TF DL-DCNN DL-BiMPM	ML-ADMA	IR-TF ML-ADMA	DL-CRNN DL-MALSTM
R5	DL-DCNN	ML-ADMA	DL-MALSTM DL-CRNN	DL-CRNN DL-MALSTM	–
R6	DL-BiMPM	DL-CRNN	–	–	–
R7	DL-CRNN DL-MALSTM	DL-MALSTM	–	–	–

*TS is the number of training projects.

We can see that four of the deep learning based approaches (i.e., *DL-DCNN*, *DL-MALSTM*, *DL-BiMPM*, and *DL-CRNN*) are sensitive to the training size. Specifically, their performance undergoes a noticeable improvement with the increase of the training data, especially when the number of training projects increase from 20 to 100. For example, although *DL-BiMPM* is one of the best two approaches with 300 projects as the training data, it is low in the ranking with only 20 training projects, indicating one should be careful in applying this approach when few training data is available. This is not surprising since deep learning based approaches are commonly known as heavily reliance on data.

We also notice that the another deep learning based approach, i.e., *DL-CWEIR*, does not show such a large performance variation. This might because it employs word embedding technique which has fewer parameters to be tuned thus requires less data than other deep learning based approaches. In addition, this approach also utilizes the TF-IDF similarity and reports' attributes similarity for determining the duplicates, and these two types of similarities are far less sensitive to training size.

For the three machine learning based approaches, the performance increases with the increase of the training data size at the beginning (i.e., the number of training projects increase from 20 to 50), and then almost keeps unchanged (i.e., the number of training projects increase from 50 to 300). This indicates these approaches are less sensitive to training size. For example, even with 50 crowdtesting projects as the training data, *ML-REP* can achieve comparable performance with the setup when employing 300 projects as training data. The two information retrieval based approaches undergo none or very slight performance variation, indicating they are the least sensitive to training size. Note that, the reason why *IR-DBTM* approach undergoes slight performance variation is because it needs training data to build an effective BM25F model and topic model.

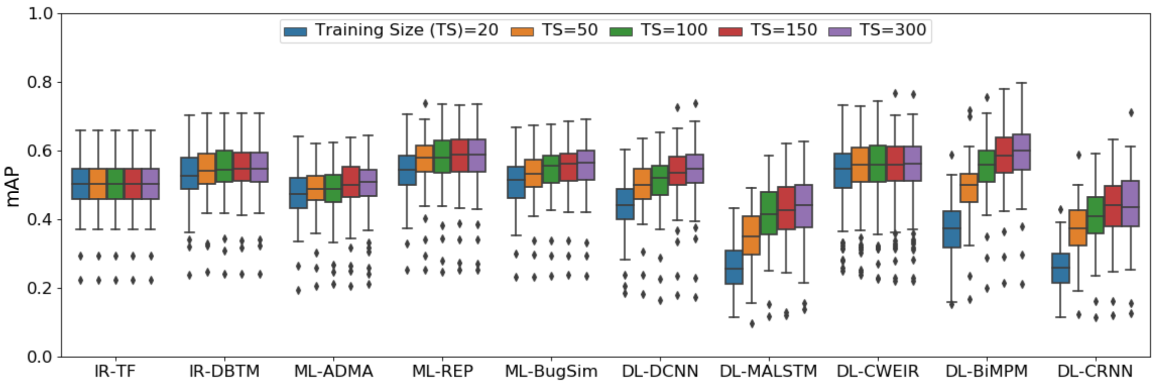

Figure 5: Performance (mAP) of ten duplicate detection approaches with different training sizes (RQ2)

When there are fewer training data (e.g., the number of training projects < 100), *ML-REP* is the best approach. If there are more training data (e.g., the number of training projects > 150), *DL-BiMPM* and *ML-REP* are the best approaches. In addition, with the increase of training data size, a larger improvement is observed on *DL-BiMPM* than *ML-REP*. These findings provide new insights and practical guidelines when applying these approaches for duplicate detection in terms of different training size.

1) Most deep learning based approaches are sensitive to the training size, while machine learning based and information retrieval based approaches are less sensitive to the training size. 2) *ML-REP* is the best approach when there are fewer training data, while *DL-BiMPM* and *ML-REP* are the best approaches with more training data.

6.3 Answering RQ3: Time Cost

Table 6 provides the model building time and prediction time with different training size. Specifically, we train the duplicate detection models with different training data (ranging from 20 to 300 projects) and then evaluate the performance of the built models on an average-sized crowdtesting project (with 32,260 report pairs).

Generally speaking, deep learning based approaches which employ Recurrent Neural Network (i.e., *DL-BiMPM*, *DL-CRNN*, and *DL-MALSTM*) are more time-consuming than other approaches in model building and duplicate detection. For example, training *DL-BiMPM* with 300 crowdtesting projects consumes around 25 hours (1513.3 minutes), while training a machine learning based approach as *ML-REP* with the same dataset requires 35.3 minutes. This is because the output of neurons in RNN is related to the previous neurons, so the network needs to be trained according to the time sequence, which makes the training of RNN more complicated and time-consuming. The deep learning based approach *DL-CWEIR* consumes much less time than other deep learning based approaches because its utilized word embedding model is a simple network structure and has fewer parameters to be tuned.

The information retrieval based and machine learning based approaches consume less time. Take the machine learning based approach *ML-REP* as an example, training the model with 300 crowdtesting projects consumes about half an hour, while conducting duplicate detection for a typical crowdtesting project requires less than 1 minute. Among these information retrieval based and machine learning based approaches, we noticed that *IR-DBTM* and *ML-AMDA* consumes more time. For *IR-DBTM*, adjusting the weight between the topic model and BM25F is time-consuming, i.e., occupying about 60% of the model training time. *ML-ADMA* employs more features than the other two machine learning based approaches, thus requires more iterations to balance these features in the final model.

Table 6: Time cost (in minutes) of the ten examined duplicate detection approaches with different training data sizes (RQ3)

	TS = 20		TS = 50		TS = 100		TS = 150		TS = 300	
Approaches	**T**	**P**	**T**	**P**	**T**	**P**	**T**	**P**	**T**	**P**
IR-TF	NA	0.1	NA	0.1	NA	0.1	NA	0.1	NA	0.1
IR-DBTM	3.3	0.7	9.5	0.6	21.7	0.6	40.5	0.5	50.0	0.5
ML-ADMA	4.7	0.1	18.2	0.1	51.3	0.1	100.8	0.1	115.2	0.1
ML-REP	2.4	0.1	7.5	0.1	17.1	0.1	31.2	0.1	35.3	0.1
ML-BugSim	1.3	0.1	4.0	0.1	8.7	0.1	15.5	0.1	17.6	0.1
DL-DCNN	3.8	0.1	11.1	0.1	25.7	0.1	46.3	0.1	52.3	0.1
DL-MALSTM	9.6	0.4	27.9	0.4	68.2	0.4	119.5	0.4	140.8	0.4
DL-CWEIR	0.1	0.4	0.1	0.4	0.1	0.6	0.1	0.7	0.1	0.7
DL-BiMPM	130.8	3.3	356.8	3.3	759.2	3.4	1329.2	3.4	1513.3	3.4
DL-CRNN	93.2	0.8	295.9	0.8	657.0	0.8	1168.5	0.8	1416.1	0.8

*TS is the number of training projects.
*T means train time and P means prediction time.

Nevertheless, since the model building step can be conducted offline, we can focus more on the duplicate detection time. Even with the most time-consuming approach *DL-BiMPM*, the duplicate detection can be done in about 5 minutes for a typical crowdtesting project. For other less time-consuming approach (e.g., *ML-REP*), the duplicate detection can be done within one minute. This implies the feasibility of applying these approaches in real-world practice.

1) The deep learning based approaches employing RNN (e.g., *DL-BiMPM*) are more time-consuming than others. 2) Information retrieval based and machine learning based approaches can conduct the duplicate detection for a typical crowdtesting project within 1 minute, and the time for *DL-BiMPM* is about 5 minutes.

7 DISCUSSION AND THREATS TO VALIDITY

The above experimental results have shown that with the learning based techniques, the duplicate reports can be recognized with high accuracy, and these techniques can be utilized to facilitate the manual duplicate detection. According to existing work [45, 47], the users prefer the actionable prediction outcomes, i.e., tagging each report with its predicted probability of being duplicates when presenting to the users for manual detection. In this way, for a candidate with higher probability, the users can treat it as the duplicate report without any consideration; on the contrary, the users can put more attention on those candidates with lower probabilities.

Construct validity of this study mainly questions the selection of the studies in the literature review. It is addressed through specifying a research protocol that defines the search terms, selection strategies, inclusion and exclusion criteria, and quality assessment. We also employ the selection verification process to let the second author review some sampled studies to further minimize the risk of exclusion of relevant studies. Besides, limited by the size of our experimental dataset, we do not examine the performance of duplicate detection with more than 300 training projects. Thus we could not conclude whether *DL-BiMPM* is better than *ML-REP* with sufficient amount of training data. We will conduct more exploration in future work.

The internal threat is the implementation of these duplicate detection approaches. We strictly follow the procedures described in the original studies, and test the implementation based on 152 test cases to ensure the correctness of duplicate detection. Furthermore, for *DL-BiMPM*, we re-use the code provided by the author, and for *DL-DCNN*, we refer to two implementations in Github to ensure the correctness of our implementation. In addition, for the attributes which are utilized in the original approaches but do not contain in our crowdtesting reports, we either use the most similar ones or ignore them in the implementation.

The external threats concern the generality of this study. Our crowdtesting dataset consists of 414 projects from one of the largest crowdtesting platforms with various domains and project sizes, which could help reduce this threats.

8 RELATED WORK

It is critical to effectively manage and triage crowdtesting reports so as to facilitate the following bug fixing process [5, 16, 48, 49, 56]. To address this challenge, Feng et al. [14, 15] and Jiang et al. [21] proposed approaches to prioritize test reports in crowdtesting. They designed strategies to dynamically select the most risky and diversified test report for inspection in each iteration. Wang et al. [44, 45, 47] proposed approaches to automatically classify crowdtesting reports. Their approaches can overcome the different data distribution among different software domains, and attain good classification results. Liu et al. [24] proposed an automatic approach to generate descriptive words of crowdtesting reports for the screenshots based on the language model and Spatial Pyramid Matching technique. Hao et al. [17] proposed CTRAS to automatically aggregate duplicates based on both textual information and screenshots, and summarize the duplicate test reports into a comprehensive and comprehensible report. Compared with the above studies, this paper focuses on the duplicate detection of crowdtesting reports.

Besides the researches mentioned in Section 3, these are several empirical studies of duplicate detection. Rocha et al. [35] conducted an empirical comparison of two duplicate detection approaches, i.e., NextBug [34] and REP [39], and results showed that the two approaches obtain similar performance when only considering the component of bug reports and short descriptions. Rakha et al. [32] conducted an empirical comparison of BM25F [33] and REP [39], and found that by using the resolution attribute of reports, the performance can be improved. Compared with these studies, this paper conducts a more thorough empirical comparison of duplicate detection approaches.

Besides the textual descriptions and attributes of reports, several researches employed other types of information to improve the duplicate detection performance. Wang et al. [52] presented an approach that uses both natural language information and execution information to compute the similarity, and designed a heuristic algorithm to combine the two similarity values. Hindle et al. [18] investigated how contextual information about software-quality attributes, software-architecture terms, and system-development topics can be used in duplicate bug report detection. Ebrahimi et al. [13] considered that only a few approaches used the execution information of the bug report, so they proposed a new approach that that automatically detects duplicate bug reports using stack traces and Hidden Markov Models. Wang et al. [46] employed the screenshot information together with textual descriptions for boosting the duplicate detection accuracy. They extracted four types of features to characterize the screenshots (i.e., image structure feature and image color feature) and the textual descriptions (i.e., TF-IDF feature and word embedding feature), and designed a hierarchical algorithm to detect duplicates based on the four similarity scores derived from the four features respectively. Although these information can improve the duplicate detection performance, they are often hard to obtain. Therefore, this work only consider the researches with textual descriptions, and future work will include more for empirical comparison.

9 CONCLUSION

This paper conducts the first empirical evaluation of ten commonly-used and state-of-the-art duplicate detection approaches in crowdtesting reports. The results show that machine learning based approach, i.e., *ML-REP*, and deep learning based approach, i.e., *DL-BiMPM*, are the best two approaches for crowdtesting reports duplicate detection, while the later one is sensitive to the size of training data. In addition, several deep learning based approaches can perform well in general sentence duplicate detection, but achieve bad performance in duplicate crowdtesting report detection due to the local bias in crowdtesting reports. We also find that the deep learning based approaches which adopt such network structure, i.e., modeling the interactive aspect of report pair, can achieve better performance. This paper provides new insights and guidelines for conducting duplicate crowdtesting reports detection in real-world crowdtesting practice.

ACKNOWLEDGMENTS

This work is supported by the National Key Research and Development Program of China under grant No.2018YFB1403400.

REFERENCES

[1] 2020. http://www.softwaretestinghelp.com/crowdsourced-testing-companies/.

[2] 2020. https://dzone.com/articles/top-10-benefits-of-crowd-testing-1.

[3] 2020. https://www.softwaretestinghelp.com/guide-to-crowdsourced-testing/.

[4] 2020. https://www.applause.com/customers/.

[5] 2020. https://dzone.com/articles/10-critical-factors-to-successfully-perform-crowds.

[6] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* 3 (2003), 993–1022. http://jmlr.org/papers/v3/blei03a.html

[7] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, Lluís Màrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton (Eds.). The Association for Computational Linguistics, 632–642. https://doi.org/10.18653/v1/d15-1075

[8] Jane Bromley, James W. Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. 1993. Signature Verification Using A "Siamese" Time Delay Neural Network. *Int. J. Pattern Recognit. Artif. Intell.* 7, 4 (1993), 669–688. https://doi.org/10.1142/S0218001493000339

[9] Amar Budhiraja, Kartik Dutta, Raghu Reddy, and Manish Shrivastava. 2018. DWEN: deep word embedding network for duplicate bug report detection in software repositories. In *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*, Michel Chaudron, Ivica Crnkovic, Marsha Chechik, and Mark Harman (Eds.). ACM, 193–194. https://doi.org/10.1145/3183440.3195092

[10] Amar Budhiraja, Raghu Reddy, and Manish Shrivastava. 2018. LWE: LDA refined word embeddings for duplicate bug report detection. In *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*, Michel Chaudron, Ivica Crnkovic, Marsha Chechik, and Mark Harman (Eds.). ACM, 165–166. https://doi.org/10.1145/3183440.3195078

[11] Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. 2005. Learning to rank using gradient descent. In *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005 (ACM International Conference Proceeding Series)*, Luc De Raedt and Stefan Wrobel (Eds.), Vol. 119. ACM, 89–96. https://doi.org/10.1145/1102351.1102363

[12] Jayati Deshmukh, K. M. Annervaz, Sanjay Podder, Shubhashis Sengupta, and Neville Dubash. 2017. Towards Accurate Duplicate Bug Retrieval Using Deep Learning Techniques. In *2017 IEEE International Conference on Software Maintenance and Evolution, ICSME 2017, Shanghai, China, September 17-22, 2017*. IEEE Computer Society, 115–124. https://doi.org/10.1109/ICSME.2017.69

[13] Neda Ebrahimi, Abdelaziz Trabelsi, Md. Shariful Islam, Abdelwahab Hamou-Lhadj, and Kobra Khanmohammadi. 2019. An HMM-based approach for automatic detection and classification of duplicate bug reports. *Inf. Softw. Technol.* 113 (2019), 98–109. https://doi.org/10.1016/j.infsof.2019.05.007

[14] Yang Feng, Zhenyu Chen, James A. Jones, Chunrong Fang, and Baowen Xu. 2015. Test report prioritization to assist crowdsourced testing. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, Bergamo, Italy, August 30 - September 4, 2015*, Elisabetta Di Nitto, Mark Harman, and Patrick Heymans (Eds.). ACM, 225–236. https://doi.org/10.1145/2786805.2786862

[15] Yang Feng, James A. Jones, Zhenyu Chen, and Chunrong Fang. 2016. Multi-objective test report prioritization using image understanding. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, ASE 2016, Singapore, September 3-7, 2016*, David Lo, Sven Apel, and Sarfraz Khurshid (Eds.). ACM, 202–213. https://doi.org/10.1145/2970276.2970367

[16] Ruizhi Gao, Yabin Wang, Yang Feng, Zhenyu Chen, and W. Eric Wong. 2019. Successes, challenges, and rethinking - an industrial investigation on crowdsourced mobile application testing. *Empirical Software Engineering* 24, 2 (2019), 537–561. https://doi.org/10.1007/s10664-018-9618-5

[17] Rui Hao, Yang Feng, James A. Jones, Yuying Li, and Zhenyu Chen. 2019. CTRAS: crowdsourced test report aggregation and summarization. In *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*, Joanne M. Atlee, Tevfik Bultan, and Jon Whittle (Eds.). IEEE / ACM, 900–910. https://doi.org/10.1109/ICSE.2019.00096

[18] Abram Hindle, Anahita Alipour, and Eleni Stroulia. 2016. A contextual approach towards more accurate duplicate bug report detection and ranking. *Empirical Software Engineering* 21, 2 (2016), 368–410. https://doi.org/10.1007/s10664-015-9387-3

[19] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[20] Alexandre Yukio Ichida, Felipe Meneguzzi, and Duncan D. Ruiz. 2018. Measuring Semantic Similarity Between Sentences Using A Siamese Neural Network. In *2018 International Joint Conference on Neural Networks, IJCNN 2018, Rio de Janeiro, Brazil, July 8-13, 2018*. IEEE, 1–7. https://doi.org/10.1109/IJCNN.2018.8489433

[21] He Jiang, Xin Chen, Tieke He, Zhenyu Chen, and Xiaochen Li. 2018. Fuzzy Clustering of Crowdsourced Test Reports for Apps. *ACM Trans. Internet Techn.* 18, 2 (2018), 18:1–18:28. https://doi.org/10.1145/3106164

[22] Tom Kenter and Maarten de Rijke. 2015. Short Text Similarity with Word Embeddings. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*, James Bailey, Alistair Moffat, Charu C. Aggarwal, Maarten de Rijke, Ravi Kumar, Vanessa Murdock, Timos K. Sellis, and Jeffrey Xu Yu (Eds.). ACM, 1411–1420. https://dl.acm.org/citation.cfm?id=2806475

[23] Barbara Kitchenham. 2004. Procedures for performing systematic reviews. *Keele, UK, Keele University* 33, 2004 (2004), 1–26.

[24] Di Liu, Xiaofang Zhang, Yang Feng, and James A. Jones. 2018. Generating descriptions for screenshots to assist crowdsourced testing. In *25th International Conference on Software Analysis, Evolution and Reengineering, SANER 2018, Campobasso, Italy, March 20-23, 2018*, Rocco Oliveto, Massimiliano Di Penta, and David C. Shepherd (Eds.). IEEE Computer Society, 492–496. https://doi.org/10.1109/SANER.2018.8330246

[25] Pengfei Liu, Xipeng Qiu, Jifan Chen, and Xuanjing Huang. 2016. Deep Fusion LSTMs for Text Semantic Matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics. https://doi.org/10.18653/v1/p16-1098

[26] Pengfei Liu, Xipeng Qiu, Yaqian Zhou, Jifan Chen, and Xuanjing Huang. 2016. Modelling Interaction of Sentence Pair with Coupled-LSTMs. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, Jian Su, Xavier Carreras, and Kevin Duh (Eds.). The Association for Computational Linguistics, 1703–1712. https://doi.org/10.18653/v1/d16-1176

[27] Ke Mao, Licia Capra, Mark Harman, and Yue Jia. 2017. A survey of the use of crowdsourcing in software engineering. *J. Syst. Softw.* 126 (2017), 57–84. https://doi.org/10.1016/j.jss.2016.09.015

[28] Tim Menzies, Suvodeep Majumder, Nikhila Balaji, Katie Brey, and Wei Fu. 2018. 500+ Times Faster than Deep Learning:(A Case Study Exploring Faster Methods for Text Mining StackOverflow). In *2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR)*. 554–563.

[29] Jonas Mueller and Aditya Thyagarajan. 2016. Siamese Recurrent Architectures for Learning Sentence Similarity. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, Dale Schuurmans and Michael P. Wellman (Eds.). AAAI Press, 2786–2792. http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12195

[30] Anh Tuan Nguyen, Tung Thanh Nguyen, Tien N. Nguyen, David Lo, and Chengnian Sun. 2012. Duplicate bug report detection with a combination of information retrieval and topic modeling. In *IEEE/ACM International Conference on Automated Software Engineering, ASE'12, Essen, Germany, September 3-7, 2012*, Michael Goedicke, Tim Menzies, and Motoshi Saeki (Eds.). ACM, 70–79. https://doi.org/10.1145/2351676.2351687

[31] Massimo Nicosia and Alessandro Moschitti. 2017. Accurate Sentence Matching with Hybrid Siamese Networks. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, Ee-Peng Lim, Marianne Winslett, Mark Sanderson, Ada Wai-Chee Fu, Jimeng Sun, J. Shane Culpepper, Eric Lo, Joyce C. Ho, Debora Donato, Rakesh Agrawal, Yu Zheng, Carlos Castillo, Aixin Sun, Vincent S. Tseng, and Chenliang Li (Eds.). ACM, 2235–2238. https://doi.org/10.1145/3132847.3133156

[32] Mohamed Sami Rakha, Cor-Paul Bezemer, and Ahmed E. Hassan. 2018. Revisiting the Performance Evaluation of Automated Approaches for the Retrieval of Duplicate Issue Reports. *IEEE Trans. Software Eng.* 44, 12 (2018), 1245–1268. https://doi.org/10.1109/TSE.2017.2755005

[33] Stephen E. Robertson, Hugo Zaragoza, and Michael J. Taylor. 2004. Simple BM25 extension to multiple weighted fields. In *Proceedings of the 2004 ACM CIKM International Conference on Information and Knowledge Management, Washington, DC, USA, November 8-13, 2004*, David A. Grossman, Luis Gravano, ChengXiang Zhai, Otthein Herzog, and David A. Evans (Eds.). ACM, 42–49. https://doi.org/10.1145/1031171.1031181

[34] Henrique Rocha, Guilherme de Oliveira, Humberto Marques-Neto, and Marco Tulio Valente. 2015. NextBug: a Bugzilla extension for recommending similar bugs. *J. Softw. Eng. Res. Dev.* 3 (2015), 3. https://doi.org/10.1186/s40411-015-0018-x

[35] Henrique Rocha, Marco Tulio Valente, Humberto Marques-Neto, and Gail C. Murphy. 2016. An Empirical Study on Recommendations of Similar Bugs. In *IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering, SANER 2016, Suita, Osaka, Japan, March 14-18, 2016 - Volume 1*. IEEE Computer Society, 46–56. https://doi.org/10.1109/SANER.2016.87

[36] Per Runeson, Magnus Alexandersson, and Oskar Nyholm. 2007. Detection of Duplicate Defect Reports Using Natural Language Processing. In *29th International Conference on Software Engineering (ICSE 2007), Minneapolis, MN, USA, May 20-26, 2007*. IEEE Computer Society, 499–510. https://doi.org/10.1109/ICSE.2007.32

[37] Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.* 45, 11 (1997), 2673–2681. https://doi.org/10.1109/78.650093

[38] Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*, Ricardo Baeza-Yates, Mounia Lalmas, Alistair Moffat, and Berthier A. Ribeiro-Neto (Eds.). ACM, 373–382. https://doi.org/10.1145/2766462.2767738

[39] Chengnian Sun, David Lo, Siau-Cheng Khoo, and Jing Jiang. 2011. Towards more accurate retrieval of duplicate bug reports. In *26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), Lawrence, KS, USA, November 6-10, 2011*, Perry Alexander, Corina S. Pasareanu, and John G. Hosking (Eds.). IEEE Computer Society, 253–262. https://doi.org/10.1109/ASE.2011.6100061

[40] Chengnian Sun, David Lo, Xiaoyin Wang, Jing Jiang, and Siau-Cheng Khoo. 2010. A discriminative model approach for accurate duplicate bug report retrieval. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1, ICSE 2010, Cape Town, South Africa, 1-8 May 2010*, Jeff Kramer, Judith Bishop, Premkumar T. Devanbu, and Sebastián Uchitel (Eds.). ACM, 45–54. https://doi.org/10.1145/1806799.1806811

[41] Yuan Tian, Chengnian Sun, and David Lo. 2012. Improved Duplicate Bug Report Identification. In *16th European Conference on Software Maintenance and Reengineering, CSMR 2012, Szeged, Hungary, March 27-30, 2012*, Tom Mens, Anthony Cleve, and Rudolf Ferenc (Eds.). IEEE Computer Society, 385–390. https://doi.org/10.1109/CSMR.2012.48

[42] Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A Deep Architecture for Semantic Matching with Multiple Positional Sentence Representations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, Dale Schuurmans and Michael P. Wellman (Eds.). AAAI Press, 2835–2841. http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/11897

[43] Fang Wang, Zhongyuan Wang, Zhoujun Li, and Ji-Rong Wen. 2014. Concept-based Short Text Classification and Ranking. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*, Jianzhong Li, Xiaoyang Sean Wang, Minos N. Garofalakis, Ian Soboroff, Torsten Suel, and Min Wang (Eds.). ACM, 1069–1078. https://doi.org/10.1145/2661829.2662067

[44] Junjie Wang, Qiang Cui, Qing Wang, and Song Wang. 2016. Towards Effectively Test Report Classification to Assist Crowdsourced Testing. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2016, Ciudad Real, Spain, September 8-9, 2016.* ACM, 6:1–6:10. https://doi.org/10.1145/2961111.2962584

[45] Junjie Wang, Qiang Cui, Song Wang, and Qing Wang. 2017. Domain Adaptation for Test Report Classification in Crowdsourced Testing. In *39th IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice Track, ICSE-SEIP 2017, Buenos Aires, Argentina, May 20-28, 2017.* IEEE Computer Society, 83–92. https://doi.org/10.1109/ICSE-SEIP.2017.8

[46] Junjie Wang, Mingyang Li, Song Wang, Tim Menzies, and Qing Wang. 2019. Images don't lie: Duplicate crowdtesting reports detection with screenshot information. *Inf. Softw. Technol.* 110 (2019), 139–155. https://doi.org/10.1016/j.infsof.2019.03.003

[47] Junjie Wang, Song Wang, Qiang Cui, and Qing Wang. 2016. Local-based active classification of test report to assist crowdsourced testing. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, ASE 2016, Singapore, September 3-7, 2016*, David Lo, Sven Apel, and Sarfraz Khurshid (Eds.). ACM, 190–201. https://doi.org/10.1145/2970276.2970300

[48] Junjie Wang, Ye Yang, Rahul Krishna, Tim Menzies, and Qing Wang. 2019. iSENSE: completion-aware crowdtesting management. In *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*, Joanne M. Atlee, Tevfik Bultan, and Jon Whittle (Eds.). IEEE / ACM, 912–923. https://doi.org/10.1109/ICSE.2019.00097

[49] Junjie Wang, Ye Yang, Tim Menzies, and Qing Wang. 2020. ISENSE2.0: Improving Completion-Aware Crowdtesting Management with Duplicate Tagger and Sanity Checker. *ACM Trans. Softw. Eng. Methodol.* 29, 4, Article 24 (July 2020), 27 pages. https://doi.org/10.1145/3394602

[50] Junjie Wang, Ye Yang, Song Wang, Yuanzhe Hu, Dandan Wang, and Qing Wang. 2020. Context-aware In-process Crowdworker Recommendation. In *Proceedings of the 42nd International Conference on Software Engineering, ICSE 2020, Seoul, Republic of Korea, May 23–29, 2020.* IEEE / ACM, 1535–1546. https://doi.org/10.1145/3377811.3380380

[51] Shuohang Wang and Jing Jiang. 2017. A Compare-Aggregate Model for Matching Text Sequences. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings.* OpenReview.net. https://openreview.net/forum?id=HJTzHtqee

[52] Xiaoyin Wang, Lu Zhang, Tao Xie, John Anvik, and Jiasu Sun. 2008. An approach to detecting duplicate bug reports using natural language and execution information. In *30th International Conference on Software Engineering (ICSE 2008), Leipzig, Germany, May 10-18, 2008*, Wilhelm Schäfer, Matthew B. Dwyer, and Volker Gruhn (Eds.). ACM, 461–470. https://doi.org/10.1145/1368088.1368151

[53] Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral Multi-Perspective Matching for Natural Language Sentences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, Carles Sierra (Ed.). ijcai.org, 4144–4150. https://doi.org/10.24963/ijcai.2017/579

[54] Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016. Sentence Similarity Learning by Lexical Decomposition and Composition. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, Nicoletta Calzolari, Yuji Matsumoto, and Rashmi Prasad (Eds.). ACL, 1340–1349. https://www.aclweb.org/anthology/C16-1127/

[55] Xinli Yang, David Lo, Xin Xia, Lingfeng Bao, and Jianling Sun. 2016. Combining Word Embedding with Information Retrieval to Recommend Similar Bug Reports. In *27th IEEE International Symposium on Software Reliability Engineering, ISSRE 2016, Ottawa, ON, Canada, October 23-27, 2016.* IEEE Computer Society, 127–137. https://doi.org/10.1109/ISSRE.2016.33

[56] Xiaofang Zhang, Yang Feng, D Liu, Z Chen, and B Xu. 2018. Research progress of crowdsourced software testing. *Journal of Software* 29, 1 (2018), 69–88.

[57] Jian Zhou and Hongyu Zhang. 2012. Learning to rank duplicate bug reports. In *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, Xue-wen Chen, Guy Lebanon, Haixun Wang, and Mohammed J. Zaki (Eds.). ACM, 852–861. https://doi.org/10.1145/2396761.2396869