POSTER: LWE: LDA refined Word Embeddings for duplicate bug report detection

Amar Budhiraja
Microsoft R&D
Hyderabad, India
ambudhir@microsoft.com

Raghu Reddy
IIIT-Hyderabad
Hyderabad, India
raghu.reddy@iiit.ac.in

Manish Shrivastava
IIIT-Hyderabad
Hyderabad, India
m.shrivastava@iiit.ac.in

ABSTRACT

Bug reporting is a major part of software maintenance and due to its inherently asynchronous nature, duplicate bug reporting has become fairly common. Detecting duplicate bug reports is an important task in order to avoid the assignment of a same bug to different developers. Earlier approaches have improved duplicate bug report detection by using the notions of word embeddings, topic models and other machine learning approaches. In this poster, we attempt to combine Latent Dirichlet Allocation (LDA) and word embeddings to leverage the strengths of both approaches for this task. As a first step towards this idea, we present initial analysis and an approach which is able to outperform both word embeddings and LDA for this task. We validate our hypothesis on a real world dataset of Firefox project and show that there is potential in combining both LDA and word embeddings for duplicate bug report detection.

1 MOTIVATION AND PROPOSED MODEL

Duplicate bug report detection has been a key problem in software maintenance in order to avoid re-assignment of same bug to different developers. Efforts have been made to assist the work of a Triager by showing a list of top-k most similar bug reports to mark duplicate bug reports. Studies have shown the potential of Latent Dirichlet Allocation (LDA) for this task [1, 4]. A recent study performed by Yang *et al.* [9] also showed potential of using word embedding techniques [3] to calculate similarity between two bug reports.

While word embeddings have a high precision (i.e. two reports which are reported as similar will have very high chances of being similar), LDA has a high recall (i.e. two reports which are reported as non-similar will have very high chances of being non-similar). In this poster, we aim to investigate this complementary relationship between word embeddings and LDA for duplicate bug report detection. Through this initial study, we demonstrate that a combined model of word embeddings and LDA can help in improving the detection of duplicate bug reports. On the same lines, we propose 'LDA refined Word Embeddings' (LWE). The basic idea of LWE is as follows. Since, LDA has a high recall, it ensures that given a report, if we find the least similar reports in the dataset, there is a very high chance that those reports are not duplicate reports of the considered report. We use this idea to perform pruning of bug reports using LDA such that least similar bug reports are not considered for flagging as duplicates. Considering the reports left after pruning from LDA, we use a word embedding model to compute similarity between the considered report and the set of un-pruned reports to show the top-k most similar reports to the Triager. In simple words, we first use an LDA model to extract top-n most similar bug reports, thereby pruning the rest. From the extracted top-n reports, we extract the final top-k most similar reports ($k < n$) by means of a word embedding model which are shown to the Triager. LWE has the following steps for detection of duplicate bug reports of an incoming reports.

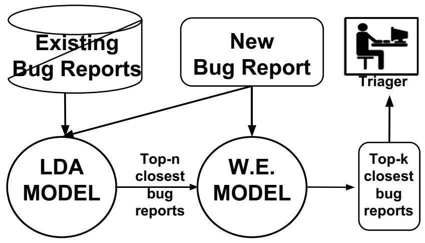


Figure 1: High level system architecture: Squares indicate input-output, circles indicate models and cylinder indicates database

(1) **Step 1:** Train LDA on bug reports database.
(2) **Step 2:** Train a word embedding model on the entire bug reports database.
(3) **Step 3:** Identify n-similar bug reports using LDA vectors from Step 1.
(4) **Step 4:** Identify top-k most similar bug reports from the n-reports identified using Step 3.

A high level system architecture is shown in Figure 1. For a new bug report, first its similarity is calculated using LDA modelling against all the existing bug reports and top-n bug reports are extracted. Using a word embedding model (W.E. Model), similarity between the new bug report and top-n extracted bug reports is computed and top-k closest bug reports are shown to the Triager.

For an i_{th} bug report, LWE can be formulated as follows,

$$LWE(i) = argmin_k \quad \mathcal{L} \cdot \mathcal{S}_{WE}(i,j) \tag{1}$$

where $\mathcal{S}_{WE}$ is a similarity function of word embeddings (e.g. cosine similarity) and $\mathcal{L}$ is defined as follows,

$$\mathcal{L} = \begin{cases} 1, & \text{if } j \in argmin_n \ \mathcal{S}_{LDA}(i,j) \\ \infty, & \text{otherwise} \end{cases} \tag{2}$$

where $\mathcal{S}_{LDA}(i,k)$ is a similarity function to compute similarity between two LDA vectors (e.g. cosine similarity). Equation 2 states the extraction of top-n bug reports using LDA and Equation 1 states extraction of top-k bug reports from the top-n reports extracted using LDA as shown in Equation 2. For the purpose of this study,

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5663-3/18/05.
https://doi.org/10.1145/3183440.3195078

Table 1: Results: Recall Rate till K=1, 10 and 20

Model	RR-1	RR-10	RR-20
LDA (400)	0.116	0.196	0.266
Skipgram (300)	0.269	0.451	0.514
LWE (LDA(400) + SG (300))	0.298	0.501	0.558

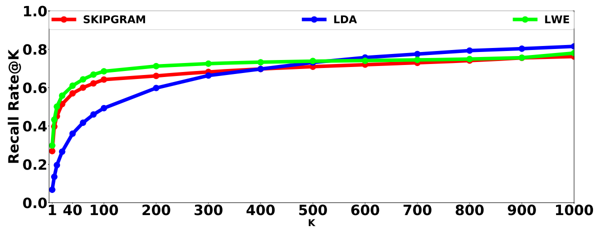
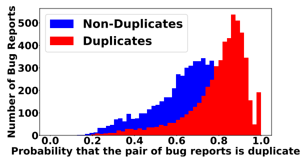
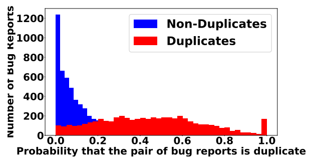


Figure 2: Graph showing 'Recall Rate till K'



(a) SKIPGRAM



(b) LDA

Figure 3: Probability distributions of duplicate bug report pairs and non-duplicate bug report pairs for Firefox project

Skipgram model [3] has been used to learn word embeddings as it has been shown to have a potential for bug similarity [9].

2 EXPERIMENTS

In the experiments section, we compare the performance of LWE with its individual components : Skipgram (word embedding model) and LDA. We also validate the claim of high precision for Skipgram and high recall for LDA.

2.1 Experimental Setup

For experiments, we have used the Mozilla Bug Report dataset [2]. The dataset contains 768335 bug reports out of which 149735 are tagged as duplicates. The dataset contains annotated duplicate bug reports. Each bug report contains an id, title (short description), description, duplicate report id, product, version, component, creation time stamp, resolution, and status. For our experiments, we only consider title and description of the bug report and combine them to consider each bug report as a document. We have used a validation set of 5000 reports and a test set of about 5000. Both the validation set and test set were chosen randomly.

Hyper-parameters: For LDA, we tuned the number of topics and passes that are run on the training corpus. Topics were varied from 100 to 400 with a step size of 50. Passes were varied from 1 to 50 with a step size of 5.[1] For Skipgram model, including LWE, we tune the following hyper-parameters : number of dimensions and epochs. We varied dimensions from 100 to 500 with a step size of 100. Epochs were varied from 1 to 50 with a step size of 1.

Performance Metric: In the literature [6–8], Recall Rate has been used and suggested for evaluation of duplicate bug report detection. *Recall rate (RR) measures the accuracy of the duplicate detection system in terms of counting the percentage of duplicates (a query which is a duplicate) for which the master bug-report is found within the top-k search results* [8]. We have used the same performance metric for our evaluations as it has been well accepted by previous works on the same problem.

[1]We have used gensim library in Python for our experiments [5].

2.2 Results

Table 1 shows the Recall Rate at 1, 10 and 20 Skipgram, LDA and LWE. Dimensions in case of Skipgram model and number of topics in case of LDA are mentioned in the brackets in column 1. It can be seen that LWE performs better than the other two approaches by a margin of approximately 3%, 5% and 5% for Recall Rate @ 1, 10 and 20 respectively. Moreover, it can be seen from Figure 2 that K = 600 LDA starts performing better whereas the word embedding model Skipgram becomes almost saturated at K=200. This validates the hypothesis that LDA has a higher recall for larger K values and word embeddings have a higher precision for smaller K values. In order to further validate this hypothesis, we take the test set of 5000 duplicate bug reports and 5000 randomly sampled 5000 non-duplicate bug reports and plot the probability of similarity, as shown in Figure 3. It can be seen that most of the mass of Skipgram for duplicate bug reports lie towards the right side whereas for non-duplicate reports the mass is more distributed through the centre. The mass for non-duplicate reports for LDA lie on the left side in the lower probability region, however the mass for duplicate reports is distributed throughout the region. This indicates that the Skipgram models gives high probability for duplicate reports (i.e. high precision) and LDA is giving low probability for non-duplicate reports (i.e. high recall). Thus, we show empirical validation of the hypothesis.

3 FUTURE WORK

In this work, we show the potential of combining LDA and word embeddings for duplicate bug report detection. As a part of future work, we plan on carrying out an in-depth investigation on why LDA has a high recall and word embeddings have high precision for this task. We plan on building a model through which we can train both LDA and word embeddings together. In the current approach, we also do not use any signals from the tagged duplicate reports, and hence, we plan on investigating if we can use these signals to improve the results such as using supervised LDA and/or supervised word embeddings.

REFERENCES

[1] Anahita Alipour et al. 2013. A contextual approach towards more accurate duplicate bug report detection. In *MSR*. IEEE.
[2] Ahmed Lamkanfi et al. 2013. The Eclipse and Mozilla Defect Tracking Dataset: a Genuine Dataset for Mining Bug Information. In *MSR*. IEEE.
[3] Tomas Mikolov et al. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
[4] Anh Tuan Nguyen et al. 2012. Duplicate bug report detection with a combination of information retrieval and topic modeling. In *ASE*. IEEE.
[5] Radim Řehůřek et al. 2010. Software Framework for Topic Modelling with Large Corpora. In *LREC Workshop on New Challenges for NLP Frameworks*.
[6] Per Runeson et al. 2007. Detection of duplicate defect reports using natural language processing. In *ICSE*. IEEE.
[7] Chengnian Sun et al. 2010. A discriminative model approach for accurate duplicate bug report retrieval. In *ICSE*. ACM.
[8] Ashish Sureka et al. 2010. Detecting duplicate bug report using character n-gram-based features. In *APSEC*. IEEE.
[9] Xinli Yang et al. 2016. Combining word embedding with information retrieval to recommend similar bug reports. In *ISSRE*. IEEE.