5 Papers

Title	Publication source	Year
Using co-change histories to improve bug localization performance	SNPD	2013
An Approach to Detecting Duplicate Bug Reports Using Natural Language and Execution Information	ICSE	2008
Combining Word Embedding with Information Retrieval to Recommend Similar Bug Reports	ISSRE	2016
Poster: DWEN: Deep Word Embedding Network for Duplicate Bug Report Detection in Software Repositories	ICSE	2018
Leveraging Syntax-Related Code for Automated Program Repair	ASE	2017

Combining Word Embedding with Information Retrieval to Recommend Similar Bug Reports

汇报人：王昭丹

CONTENTS

01. INTRODUCTION

02. DATA COLLECTION & PREPROCESSING

03. PROPOSED APPROACH

04. EXPERIMENTS AND RESULTS

INTRODUCTION

Similar Bug

Bugs handle of many common source code files.

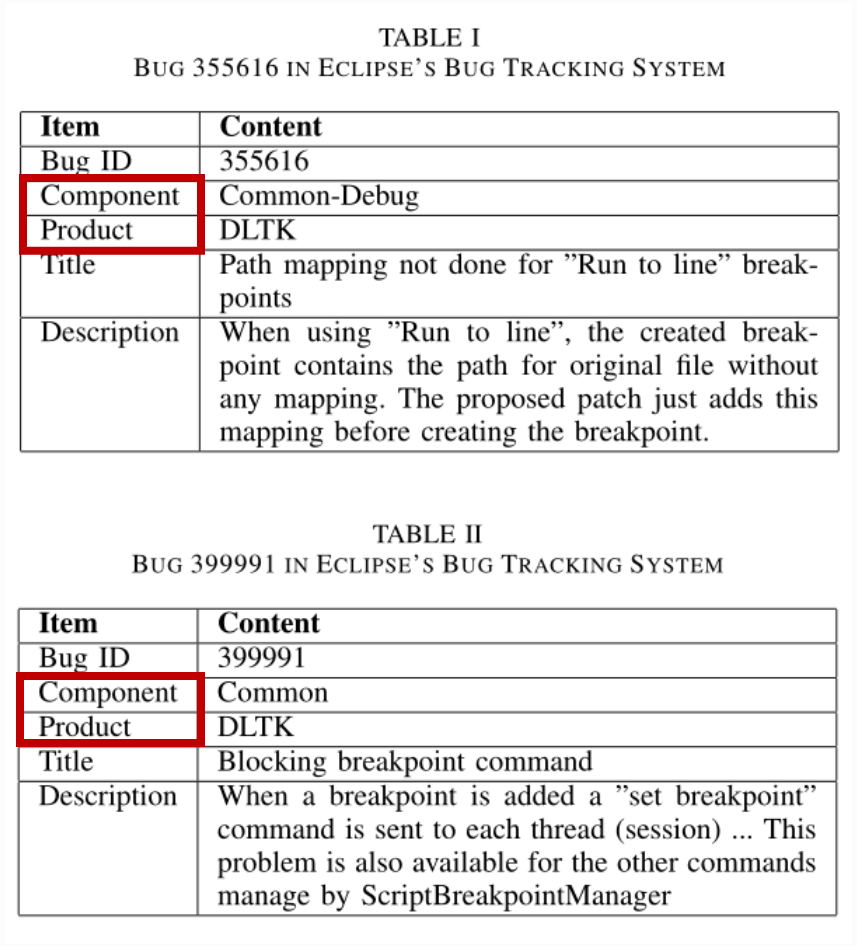


TABLE I
BUG 355616 IN ECLIPSE'S BUG TRACKING SYSTEM

Item	Content
Bug ID	355616
Component	Common-Debug
Product	DLTK
Title	Path mapping not done for "Run to line" break-points
Description	When using "Run to line", the created break-point contains the path for original file without any mapping. The proposed patch just adds this mapping before creating the breakpoint.

TABLE II
BUG 399991 IN ECLIPSE'S BUG TRACKING SYSTEM

Item	Content
Bug ID	399991
Component	Common
Product	DLTK
Title	Blocking breakpoint command
Description	When a breakpoint is added a "set breakpoint" command is sent to each thread (session) ... This problem is also available for the other commands manage by ScriptBreakpointManager

- Not **Duplicate** !
- Same product
- Same component
- Handle the same file

"core/plugins/org.eclipse.dltk.debug/src/org/eclipse/dltk/internal/debug/core/model/ScriptBreakpointManager.java"

INTRODUCTION

distributional hypothesis

↓

distributional semantic models (DSMs)

↓

count-based models

↓

based on neural network，word embedding

Word Embedding

- **CBOW** model predicts the current word based on its context
- **Skip-gram** model predicts the surrounding words given the current word

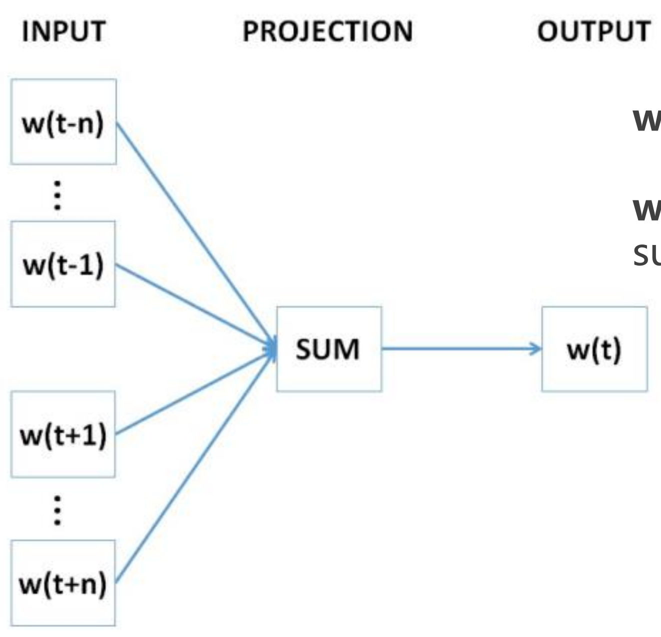


Fig. 1. The Architectures of CBOW Models

w(t) represents the current word

w(t±i) (i=1,2,...,n) represents the surrounding words of w(t)

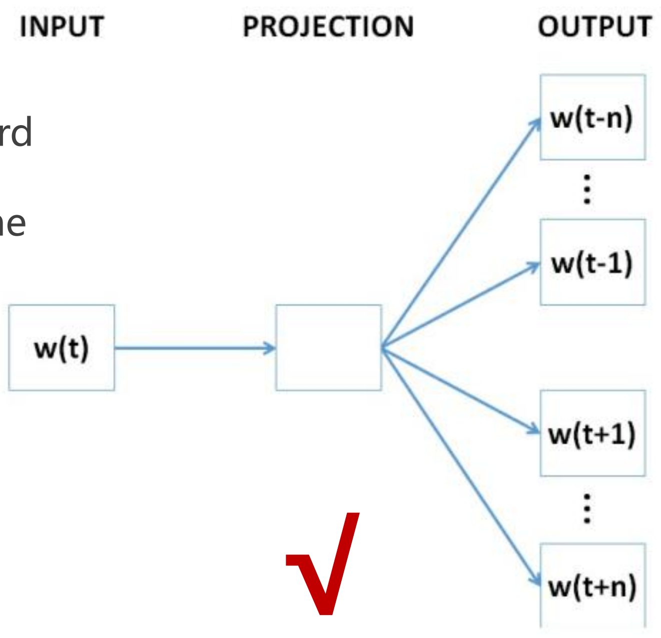


Fig. 2. The Architectures of Skip-gram Models

INTRODUCTION

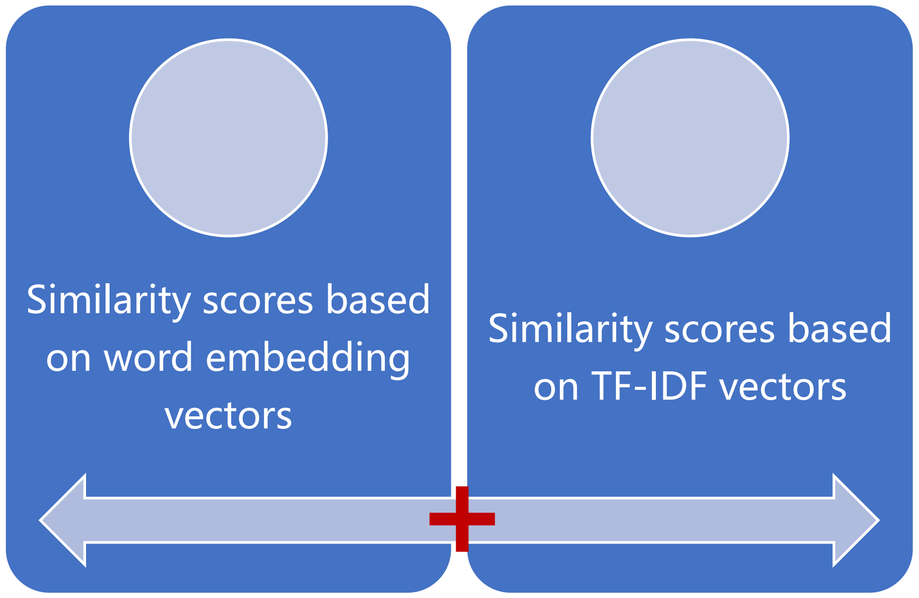
Word Embedding

- focus more on the relationship of words considering the context they appear

Complementary

Traditional IR Techniques

- focus more on the relationship of different documents in the whole corpus, such as TF-IDF


Similarity scores based on word embedding vectors
Similarity scores based on TF-IDF vectors
+

Compute and combine two similarity scores

DATA COLLECTION & PREPROCESSING

Collect bug reports from Eclipse and Mozilla, bug ids range from [1,400000]

Extract its title and description, the product and component information

Combine its title and description into a single document

Preprocess the document

Integrate the product and component information as a set

Preprocess the document

- extract all the terms (i.e., words) from the document
- remove stop words, numbers, punctuation marks and other non-alphabetic characters
- use the Snowball stemmer to transform the remaining terms to their root forms

Extracting Information From Bug Reports

- **Note:** use bug reports submitted at least 3 years ago
- **For the product and component information:** directly extract them from two fields of bug reports, product field and component field
- each bug corresponds to a set, **set = {p, c}**, p - product information, c - component information

DATA COLLECTION & PREPROCESSING

Collect 3,838,708 commit logs of Eclipse and Mozilla from GitHub

Use the approach proposed by Sliwerski et al. to **extract** bug ids from commit messages

Union the committed file lists which correspond to the same bug id

Link bug reports and committed file lists according to the bug ids

Building Ground Truth

- use the method proposed by Rocha et al. to label similar bugs
- given two **bugs x** and **y**, calculate the ratio of mutually committed files —— Mutual(x, y)

$$Mutual(x,y) = \frac{|F_x \bigcap F_y|}{min(|F_x|, |F_y|)}$$

Fx and **Fy** represents the committed file lists of x and y

- set a threshold for Mutual(x, y)
- compare each pair of bugs and label all the similar bugs as the ground truth

- **13,337,653** pairs of similar bugs in **Eclipse**
- **10,596,675** pairs of similar bugs in **Mozilla**

PROPOSED APPROACH —— OVERALL FRAMEWORK

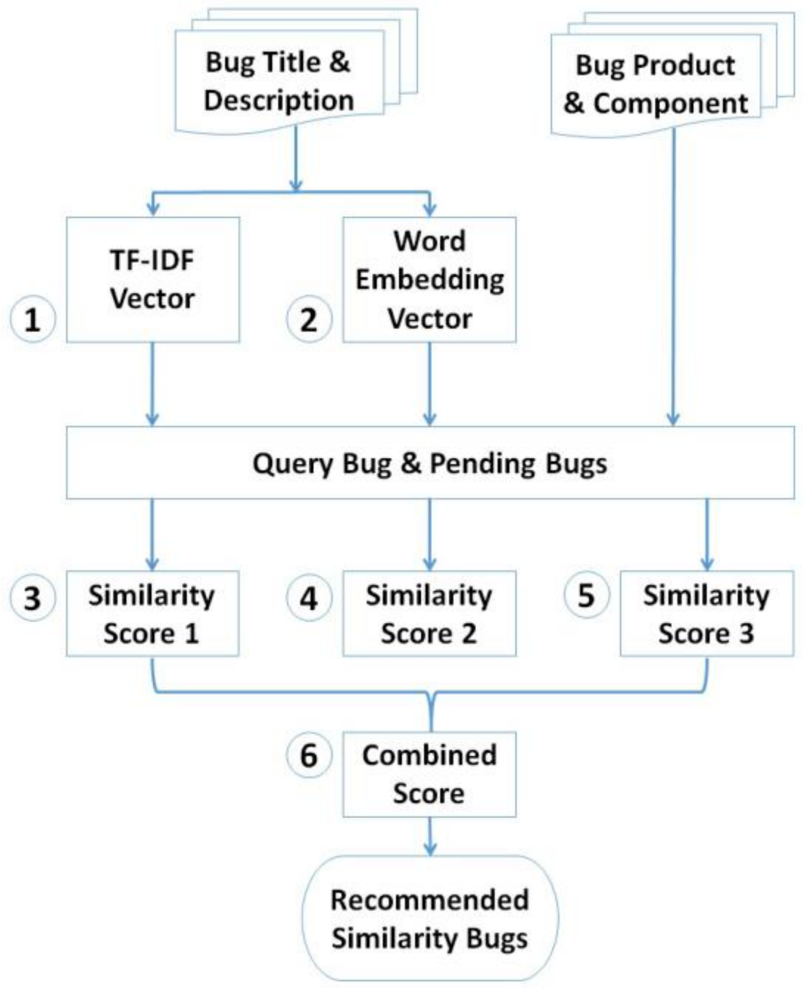


Fig. 3. The Overall Framework of Our Approach.

Four components

- TF-IDF component
- word embedding component
- product & component information component
- combination component

PROPOSED APPROACH

TF-IDF Vectors

Given a term **t** and a document **d**, define TF and IDF as follows:

$$TF(t,d) = \frac{Number\ of\ times\ t\ appears\ in\ d}{Number\ of\ terms\ in\ d}$$

$$IDF(t) = \log \frac{Total\ number\ of\ documents}{Number\ of\ documents\ that\ contain\ t+1}$$

$$\downarrow$$

$$\text{TF-IDF}(t,d) = TF(t,d) \times IDF(t)$$

Given two TF-IDF vectors **v1 v2**

cosine similarity

$$\downarrow$$

$$Score_1 = \frac{v_1 \cdot v_2}{|v_1| * |v_2|}$$

Word Embedding Vectors

Given a word **w**, the set of the surrounding context words of w as **Cw**.

The objective function **J** of a skip-gram model:

$$J = \sum_{i=1}^{n} \sum_{w_j \in C_{w_i}} \log \underline{p(w_j|w_i)}$$

vector representation of the word w

$$p(w_j \in C_{w_i}|w_i) = \frac{exp(\boxed{v_{w_j}^T} v_{w_i})}{\sum_{w \in \boxed{W}} exp(v_w^T v_{w_i})}$$

vocabulary of all words

given a document matrix that has **n** rows

the i-th row of the matrix as **ri**

transformed document vector **vd**

$$\downarrow$$

$$v_d = \frac{\sum_i r_i}{n}$$

cosine similarity

$$\longrightarrow$$

Score2

PROPOSED APPROACH

Product & Component Information

Almost all similar bugs are in the same product and component.

Given two bugs, denote their corresponding sets as **set1** and **set2**. The similarity score Score3 is:

$$Score_3 = \frac{|set_1 \bigcap set_2|}{|set_1 \bigcup set_2|}$$

Similar Bug Recommendation

- **Score1** is generated based on TF-IDF vectors, which focus more on relationship of different documents in the whole corpus.
- **Score2** is generated based on word embedding vectors, which focus more on the relationship of words considering the context they appear.
- **Score3** can be seen as a filter.

$$Score = (Score_1 + Score_2) \times Score_3$$

EXPERIMENTS AND RESULTS

Evaluation Metrics

- **Recall-rate@k:** checks whether a top-k recommendation is useful.
- **MAP (Mean Average Precision):** the mean of the Average Precision (AvgP) values obtained for all the evaluation queries.
- **MRR (Mean inverse Rank):** the mean of the Reciprocal Rank (RR) values obtained for all the evaluation queries.

Experimental Settings

- All bugs in our datasets are regarded as query bugs once.
- Given a query bug **q**, only recommend its potential similar bugs whose bug ids are less than that of q.
- **Word embedding technique** is applied by using the python package gensim1.

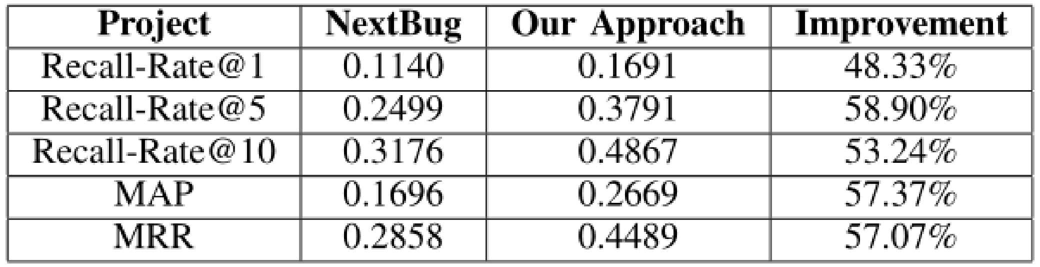
EXPERIMENTS AND RESULTS

How effective is the approach to recommend similar bugs

- Compare the approach against NextBug
- Set the threshold as 0.3 for NextBug
- **Improvement**: nearly **60%** for the Eclipse dataset and nearly **85%** for the Mozilla dataset in terms of all the metrics

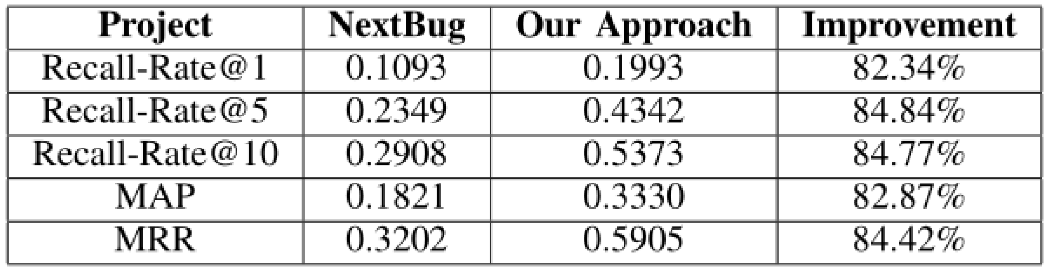
1

TABLE III
PERFORMANCE OF OUR APPROACH COMPARED WITH NEXTBUG FOR THE ECLIPSE DATASET



Project	NextBug	Our Approach	Improvement
Recall-Rate@1	0.1140	0.1691	48.33%
Recall-Rate@5	0.2499	0.3791	58.90%
Recall-Rate@10	0.3176	0.4867	53.24%
MAP	0.1696	0.2669	57.37%
MRR	0.2858	0.4489	57.07%

TABLE IV
PERFORMANCE OF OUR APPROACH COMPARED WITH NEXTBUG FOR THE MOZILLA DATASET



Project	NextBug	Our Approach	Improvement
Recall-Rate@1	0.1093	0.1993	82.34%
Recall-Rate@5	0.2349	0.4342	84.84%
Recall-Rate@10	0.2908	0.5373	84.77%
MAP	0.1821	0.3330	82.87%
MRR	0.3202	0.5905	84.42%

EXPERIMENTS AND RESULTS

Does the combined similarity score generated by the approach works better than the three individual similarity scores?

- Compare the approach with three incomplete versions – **Sub-1**, **Sub-2** and **Sub-3**.
- For Sub-1, only use the first similarity score, TF-IDF vectors;
- For Sub-2, only use the second similarity score, word embedding vectors;
- For Sub-3, only use the third similarity score, product and component information;

2

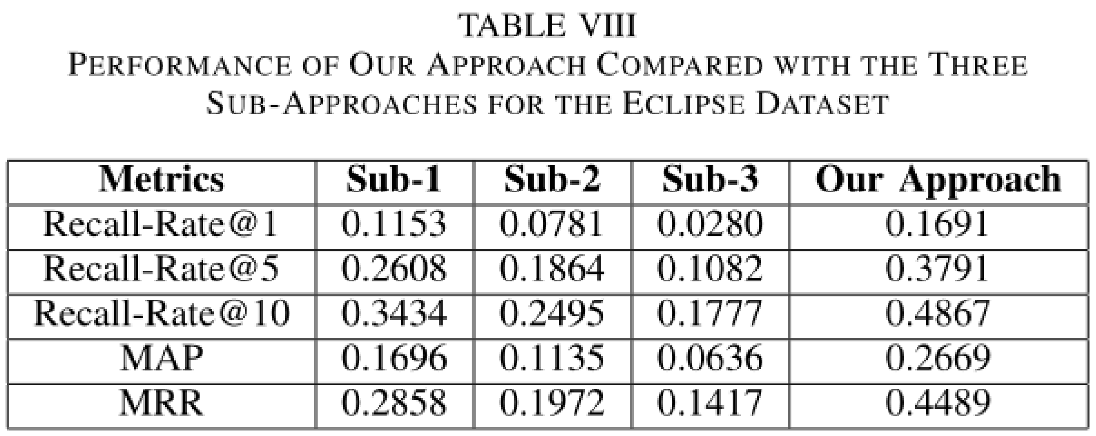


TABLE VIII
PERFORMANCE OF OUR APPROACH COMPARED WITH THE THREE SUB-APPROACHES FOR THE ECLIPSE DATASET

Metrics	Sub-1	Sub-2	Sub-3	Our Approach
Recall-Rate@1	0.1153	0.0781	0.0280	0.1691
Recall-Rate@5	0.2608	0.1864	0.1082	0.3791
Recall-Rate@10	0.3434	0.2495	0.1777	0.4867
MAP	0.1696	0.1135	0.0636	0.2669
MRR	0.2858	0.1972	0.1417	0.4489

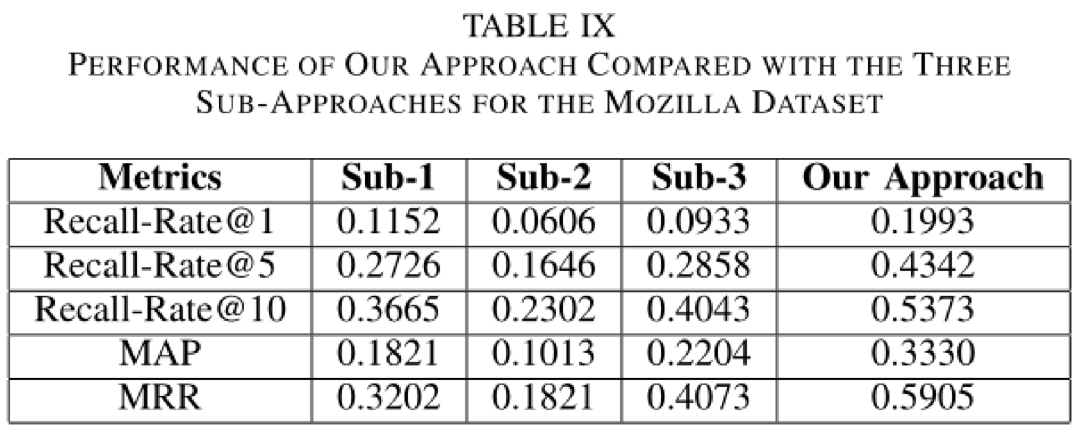
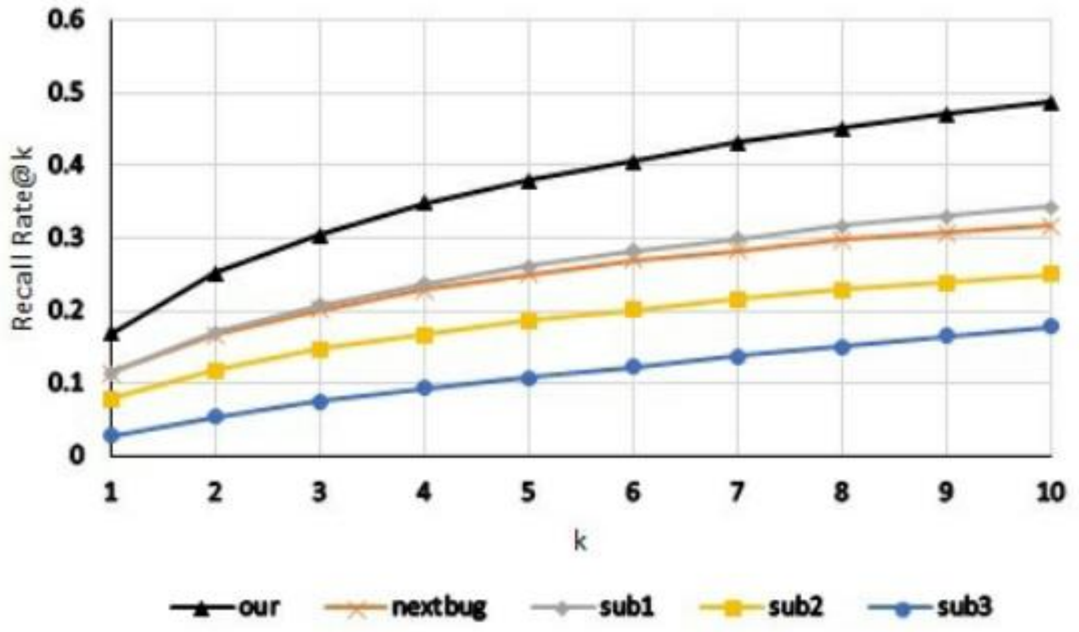


TABLE IX
PERFORMANCE OF OUR APPROACH COMPARED WITH THE THREE SUB-APPROACHES FOR THE MOZILLA DATASET

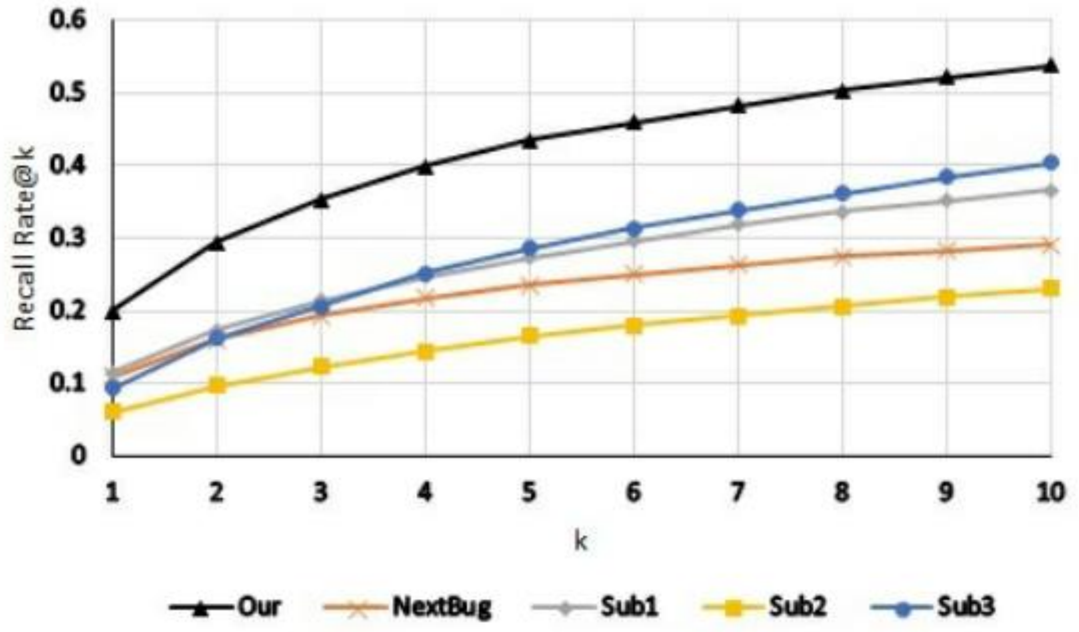
Metrics	Sub-1	Sub-2	Sub-3	Our Approach
Recall-Rate@1	0.1152	0.0606	0.0933	0.1993
Recall-Rate@5	0.2726	0.1646	0.2858	0.4342
Recall-Rate@10	0.3665	0.2302	0.4043	0.5373
MAP	0.1821	0.1013	0.2204	0.3330
MRR	0.3202	0.1821	0.4073	0.5905

EXPERIMENTS AND RESULTS

Recommendation performance of the approach steadily rises with the increase of k and is better than those of the other four approaches by a large margin.



(a) Eclipse



(b) Mozilla

Fig. 4. The recall-rate@k of five approaches when varying the parameter k in top-k recommendation on the Eclipse and Mozilla datasets

学 习 进 展 & 暑 期 计 划

感 谢 您 的 聆 听

汇报人：王昭丹

Extracting Information From Bug Reports

1. first collect bug reports，bug ids range from [1,400000]，two large open-source projects，Eclipse and Mozilla
2. For each bug report, we extract its title and description as well as the product and component information.
3. combine its title and description into a single document
4. preprocess the document with the following steps
   a. extract all the terms (i.e., words) from the document
   b. remove stop words, numbers, punctuation marks and other non-alphabetic characters
   c. use the Snowball stemmer [20] to transform the remaining terms to their root forms
5. For the product and component information, directly extract them from two fields of bug reports
6. integrate the two kinds of information as a set，set = {p, c}，p - product information and c - component information

Building Ground Truth

1. collect in total 3,838,708 commit logs of Eclipse and Mozilla from GitHub
2. use the approach proposed by Sliwerski et al. to extract bug ids from commit messages
3. union the committed file lists which correspond to the same bug id
4. link bug reports and committed file lists according to the bug ids