三篇论文

SCIENCE AND TECHNOLOGY

《An Empirical Study on Heterogeneous Defect Prediction Approaches》	2020	TSE
《Classifying Mobile Applications Using Word Embeddings》	2021	TOSEM
《From word embeddings to document similarities for improved information retrieval in software engineering》	2016	ICSE

GRADUATION DEFENSE

An Empirical Study on Heterogeneous Defect Prediction Approaches

2020 TSE

HDP(Heterogeneous Defect Prediction)

WPDP

- *Typical CPDP context,* in which the source and target projects from a prediction combination have the same metric set.

- *Typical HDP context,* in which the source and target projects from a prediction combination have different metric sets.

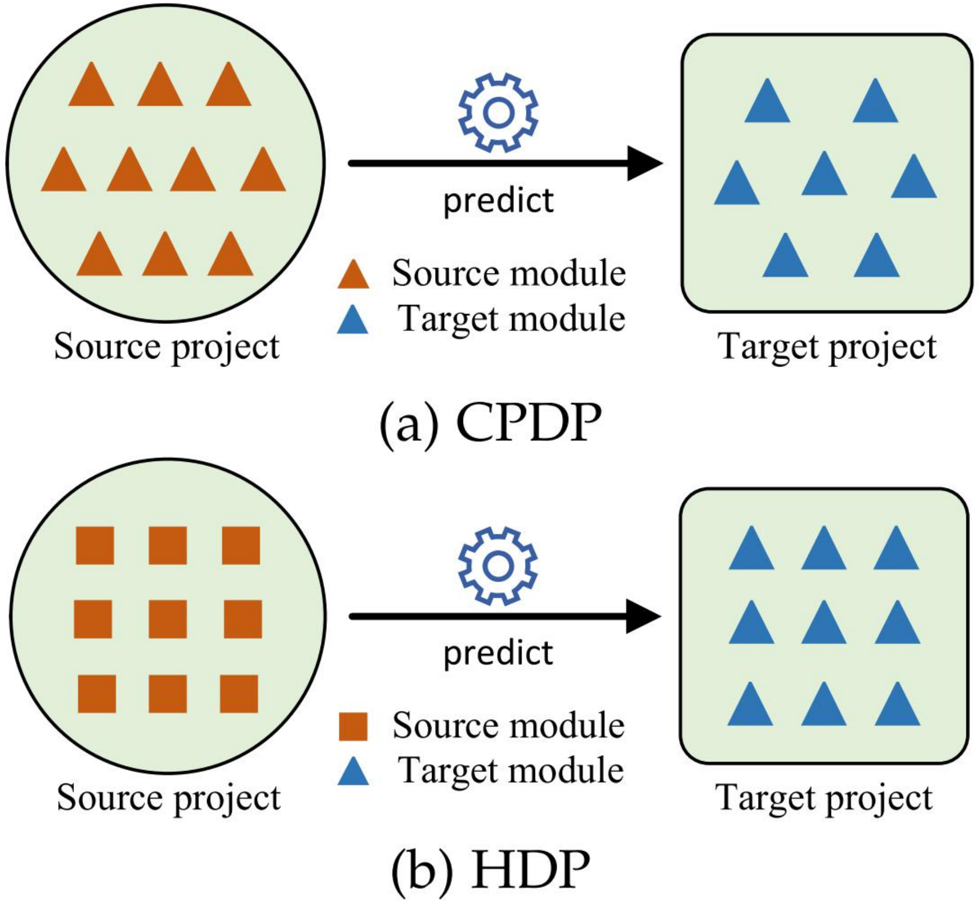


Fig. 1. The CPDP and HDP prediction combinations.

An Empirical Study on Heterogeneous Defect Prediction Approaches

Analysis

2020 TSE

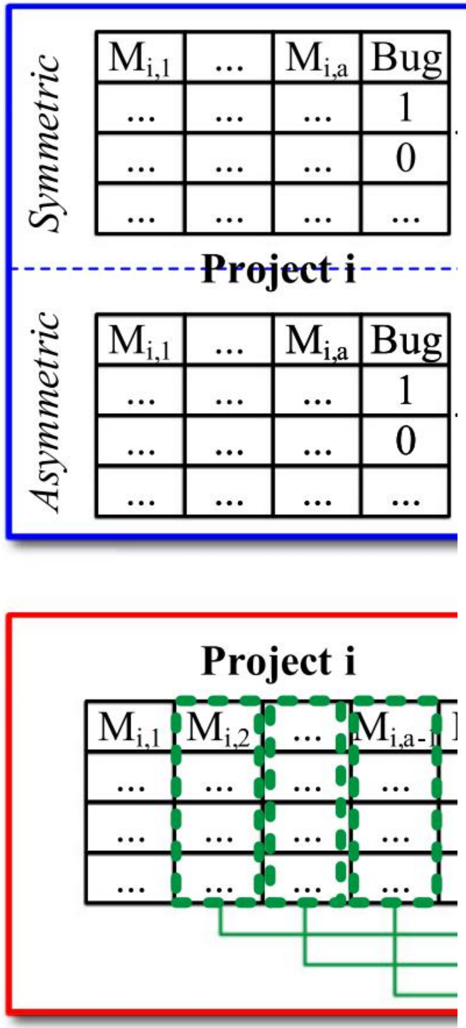


TABLE 4
The Categories of HDP Approaches

Approach	Metric transformation		Metric selection
	Symmetric	Asymmetric	
CPDP-IFS[30]	√	×	×
CCA+[28]	√	×	×
HDP-KS[40]	×	×	√
CCT-SVM[41]	√	×	×
FMT[42]	×	×	√
EMKCA[43]	×	√	×
CTKCCA[44]	√	×	×
CLSUP[46]	×	√	×
MSMDA[48]	√	×	×

Fig. 2. The processes of metric transformation-based and metric selection-based HDP approaches.

ommon latent
rce and target
formation maps

An Empirical Study on Heterogeneous Defect Prediction Approaches

2020 TSE

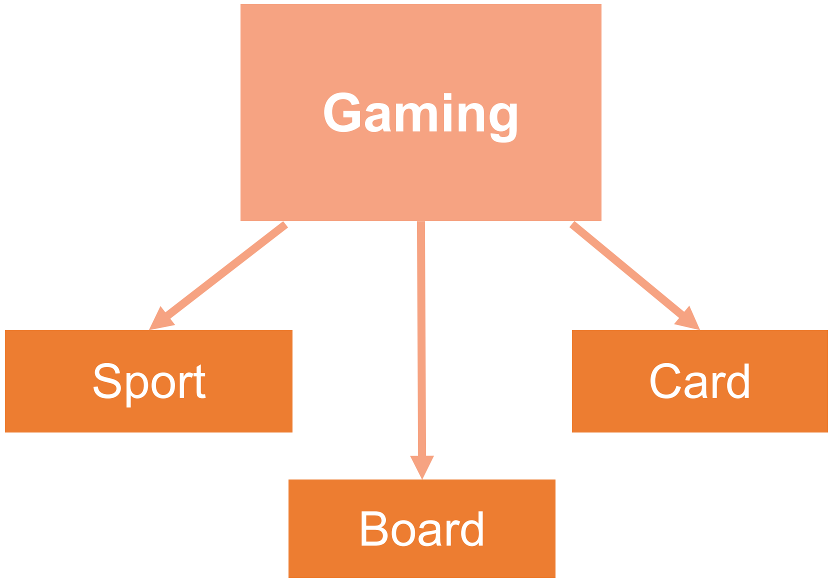
TABLE 5
The summary of GQM

Goal		Question	Metric
Purpose:	Understanding	**RQ1:** Which of the metric selection-based and metric transformation-based HDP approaches have the best prediction performance?	*AUC*, *F-measure*, *G-measure*, *MCC*, *Rankscore*
Issue:	the progress	**RQ2:** How about the effects of such improvements (i.e., handling class imbalance problems and utilizing mixed project data) on HDP approaches' performance?	*AUC*, *F-measure*, *G-measure*, *MCC*
Object:	of HDP approaches	**RQ3:** Are the HDP approaches feasible for cross-project defect prediction in which the source and target projects have the same metric set?	*AUC*, *F-measure*, *G-measure*, *MCC*, *Rankscore*
Viewpoint:	from the viewpoint of prediction performance		
Context:	within typical HDP and CPDP contexts.		

Classifying Mobile Applications Using Word Embeddings

2021 TOSEM

Motivation



The problem of app classification will persist as long as the number of apps in app stores continues to grow.

The search engines of popular app stores used to provide adequate accessibility to apps. However, after the explosive growth in the mobile app market in recent years as well as the constant changes in app store ranking policies, relying on a general keyword search can no longer guarantee equal access to apps.

Classifying Mobile Applications Using Word Embeddings

2021 TOSEM

Word2Vec
GloVe
FastText

NB
SVM
RF
DT

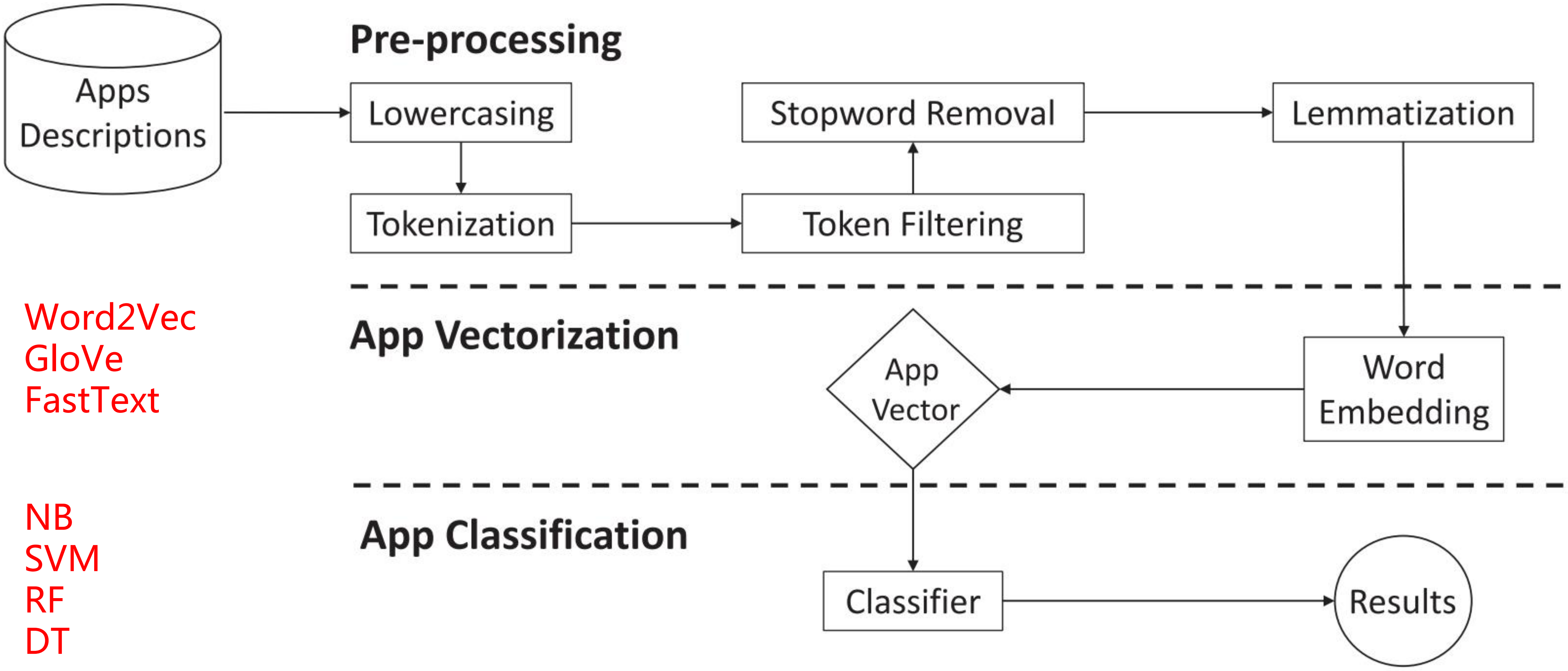


Fig. 4. The main steps of the proposed approach.

Classifying Mobile Applications Using Word Embeddings

2021 TOSEM

Approach	Classifier	Education categories			Education sub_categories			Health apps		
		P	R	F_2	P	R	F_2	P	R	F_2
VSM	NB	0.53	0.56	0.55	0.28	0.35	0.33	0.5	0.57	0.55
	AdaBoost	0.37	0.39	0.38	0.28	0.25	0.25	0.33	0.42	0.39
	Random Forest	0.67	0.65	0.65	0.46	0.45	0.45	0.55	0.59	0.58
	KNN	0.65	0.62	0.62	0.56	0.51	0.51	0.64	0.6	0.6
	SVM	0.71	0.69	0.69	0.57	0.5	0.51	0.64	0.66	0.65
	Decision Trees	0.56	0.51	0.51	0.44	0.38	0.39	0.52	0.51	0.51
	Logistic Regression	0.68	0.67	0.67	0.44	0.45	0.44	0.56	0.61	0.59
	Average	0.59	0.58	0.58	0.43	0.41	0.41	0.53	0.56	0.55
LDA	NB	0.27	0.32	0.3	0.15	0.12	0.12	0.41	0.27	0.28
	AdaBoost	0.42	0.35	0.36	0.14	0.2	0.18	0.32	0.29	0.29
	Random Forest	0.49	0.41	0.42	0.29	0.28	0.28	0.33	0.31	0.31
	KNN	0.38	0.35	0.35	0.26	0.25	0.25	0.34	0.28	0.29
	SVM	0.35	0.41	0.39	0.2	0.29	0.26	0.33	0.38	0.36
	Decision Trees	0.44	0.38	0.39	0.29	0.25	0.25	0.3	0.27	0.27
	Logistic Regression	0.35	0.4	0.38	0.23	0.31	0.28	0.33	0.39	0.37
	Average	0.38	0.37	0.37	0.22	0.24	0.23	0.33	0.31	0.31
GloVe 300	NB	0.7	0.68	0.68	**0.67**	**0.63**	**0.63**	0.72	0.66	0.67
	AdaBoost	0.69	0.67	0.67	0.4	0.38	0.38	0.54	0.55	0.54
	Random Forest	0.73	0.71	0.71	0.56	0.53	0.53	0.7	0.71	0.7
	KNN	0.74	0.72	0.72	0.66	0.58	0.59	0.81	0.78	0.78
	SVM	**0.85**	**0.84**	**0.84**	0.6	0.57	0.57	**0.83**	**0.8**	**0.8**
	Decision Trees	0.63	0.59	0.59	0.46	0.44	0.44	0.6	0.58	0.58
	Logistic Regression	0.82	0.8	0.8	0.57	0.54	0.54	0.79	0.78	0.78
	Average	0.73	0.71	0.71	0.56	0.52	0.53	0.71	0.69	0.69
Word2Vec	NB	0.74	0.69	0.69	0.62	0.52	0.53	0.68	0.63	0.63
	AdaBoost	0.55	0.51	0.51	0.21	0.22	0.21	0.4	0.37	0.37
	Random Forest	0.67	0.68	0.67	0.52	0.48	0.48	0.62	0.65	0.64
	KNN	0.64	0.64	0.64	0.53	0.48	0.48	0.64	0.58	0.59
	SVM	0.67	0.68	0.67	0.43	0.45	0.44	0.59	0.64	0.62
	Decision Trees	0.47	0.45	0.45	0.34	0.28	0.29	0.5	0.48	0.48
	Logistic Regression	0.66	0.66	0.66	0.41	0.45	0.44	0.54	0.62	0.6
	Average	0.62	0.61	0.61	0.43	0.41	0.41	0.56	0.56	0.56

From word embeddings to document similarities for improved information retrieval in software engineering

2016 ICSE

Lexical Gap

Bug ID: 384108
Summary: JUnit view icon no longer shows progress while executing tests
Description: Before I upgraded to Juno this morning I used to happily run tests with the JUnit view minimized, and enjoy seeing the progress of the tests on it. Now I don't see any change on the icon until if passes (where a green check appears) or fails (where a red X appears) ...

Figure 1: Eclipse bug report 384108, with keywords in red.

```
public class PartServiceImpl implements EPartService {
   ...
   private void recordStackActivation(MPart part) {...
      MPlaceholder placeholder = part.getCurSharedRef();
   ... }
   ...
   private void adjustPlaceholder(MPart part) {...
      MPlaceholder placeholder = part.getCurSharedRef();
   ... }
... }
```

Figure 2: Code from PartServiceImpl.java, with keywords in blue.

From word embeddings to document similarities for improved information retrieval in software engineering

2016 ICSE

Skip-gram Model

One-vocabulary setting

A single vocabulary is created to contain both words and tokens.

Two-vocabulary setting

Two vocabularies are created. One is used for words appearing in the natural language text and the other is used for tokens appearing in the code.

From word embeddings to document similarities for improved information retrieval in software engineering

2016 ICSE

Learning Word Embeddings on Software Documents :

Skip-gram Model

API documents, tutorials, reference documents

1

Heuristic Mapping of Tokens to Words

A context must be created with a `Drawable`, usually an SWT `Canvas`, on which OpenGL renders its scenes.

The application uses `GLScene`, which is a utility class for displaying OpenGL scenes.
The `GLScene` class is similar to SWT's `Canvas`.

`GLScene` uses the entire area of the canvas for drawing. In the constructor, a new SWT `Canvas` is created. This is the canvas that is associated with a `GLContext` instance.

WorkbenchWindow

2

Semantic Pairings between Text and Code

The `GLScene` class is similar to SWT's `Canvas`. However, rather than using a `GC` to draw on it, its content is rendered by OpenGL commands. This is achieved by associating a `GLContext` with an SWT `Canvas` and making it the current context whenever a scene is rendered by the commands defined in the `drawScene` method.

```
void connect(IStreamsProxy streamsProxy)
```

Connects this console to the given streams proxy. This associates the standard in, out, and error streams with the console. Keyboard input will be written to the given proxy.

Figure 9: Example of semantically related text and code, from API documents.

code, from the same tutorial.

From word embeddings to document similarities for improved information retrieval in software engineering

2016 ICSE

Table 3: The vocabulary size.

Word embeddings trained on:	Vocabulary size
one-vocabulary setting	21,848
two-vocabulary setting	25,676

Table 4: Number of word pairs.

Approach	# of word pairs
One-vocabulary embeddings	238,612,932
Two-vocabulary embeddings	329,615,650
SEWordSim [42]	5,636,534
SWordNet [45]	1,382,246

From word embeddings to document similarities for improved information retrieval in software engineering

2016 ICSE

Document Similarities

$$sim(w_t, w_u) = cos(\mathbf{w}_t, \mathbf{w}_u) = \frac{\mathbf{w}_t^T \mathbf{w}_u}{\|\mathbf{w}_t\| \|\mathbf{w}_u\|} \quad (3)$$

$$sim(w, T) = \max_{w' \in T} sim(w, w') \quad (4)$$

$$sim(T \to S) = \frac{\sum_{w \in T} sim(w, S) * idf(w)}{\sum_{w \in T} idf(w)} \quad (5)$$

$$sim(T, S) = sim(T \to S) + sim(S \to T) \quad (6)$$

$$P(T \to S) = \{w \in T | sim(w, S) \neq 0\}$$

$$sim(T \to S) = \frac{\sum_{w \in P(T \to S)} sim(w, S)}{|P(T \to S)|} \quad (7)$$

From word embeddings to document similarities for improved information retrieval in software engineering

2016 ICSE

Table 10: LR+WE[1] results obtained using the enhanced vs. the original Skip-gram model.

Project	Metric	LR ϕ_1-ϕ_8	Enhanced Skip-gram ϕ_1-ϕ_6	Original Skip-gram ϕ_1-ϕ_8
Eclipse Platform UI	MAP	0.37	0.40	0.40
	MRR	0.44	0.46	0.46
JDT	MAP	0.35	0.42	0.42
	MRR	0.43	0.51	0.51
SWT	MAP	0.36	0.38	0.37
	MRR	0.43	0.45	0.44
Birt	MAP	0.19	0.21	0.21
	MRR	0.24	0.27	0.27

Table 11: Comparison between the new text similarity function (LR+WE[1]) and the original similarity function ($\mathbf{LR+WE^1_{ori}}$).

Project	Metric	LR ϕ_1-ϕ_8	LR+WE[1] ϕ_1-ϕ_6	$LR+WE^1_{ori}$ ϕ_7-ϕ_8
Eclipse Platform UI	MAP	0.37	0.40	0.37
	MRR	0.44	0.46	0.43
JDT	MAP	0.35	0.42	0.36
	MRR	0.43	0.51	0.45
SWT	MAP	0.36	0.38	0.37
	MRR	0.43	0.45	0.44
Birt	MAP	0.19	0.21	0.20
	MRR	0.24	0.27	0.25