Compositional Vector Space Models for Improved Bug Localization

Supplement Presentation

2018-5-4

Lottie Lu, @Computer Graphix Lab, University of Seoul

Term frequency and Inverse document frequency (tf-idf).

- The standard tf-idf scheme assigns a weight to a term **t** in a document **d** according to the formula:

$$\text{weight}(t, d) = \text{tf}(t, d) \times \text{idf}(t, D)$$

Where t, d, D, tf (t, d), idf (t,D) correspond to a term, a document, a corpus (i.e., a set of documents), the frequency of t in d, and the inverse document frequency of t in D, respectively.

3. Variants of the TF-IDF Weighting Scheme Search-Based Composition Engine

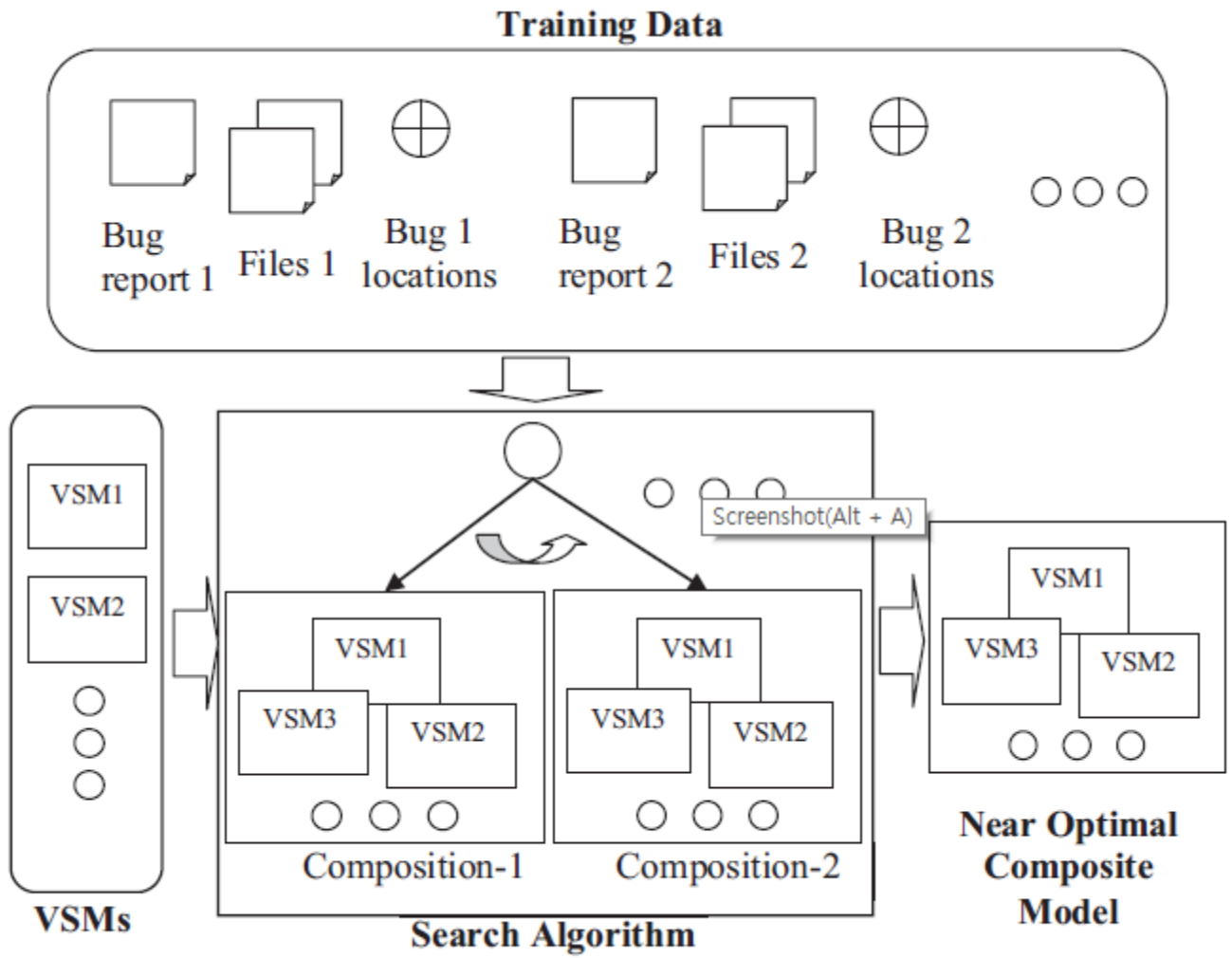
- The tf-idf weight for a term in a document is the product of its **term frequency score** and its **inverse document frequency score**.
- **Inverse document frequency:** (idf) is a measure of whether a term is common or rare in the documents of a corpus.

- There are many **variants of the standard tf-idf** weighting scheme, depending on how the tf and idf are measured.

4.Search-Based Composition Engine

- Our search-based bug localization process is composed of two phases:
- **A. Training Phase**
- **B. Deployment Phase**

The two phases are illustrated in Figure 2.



(a) Training Phase

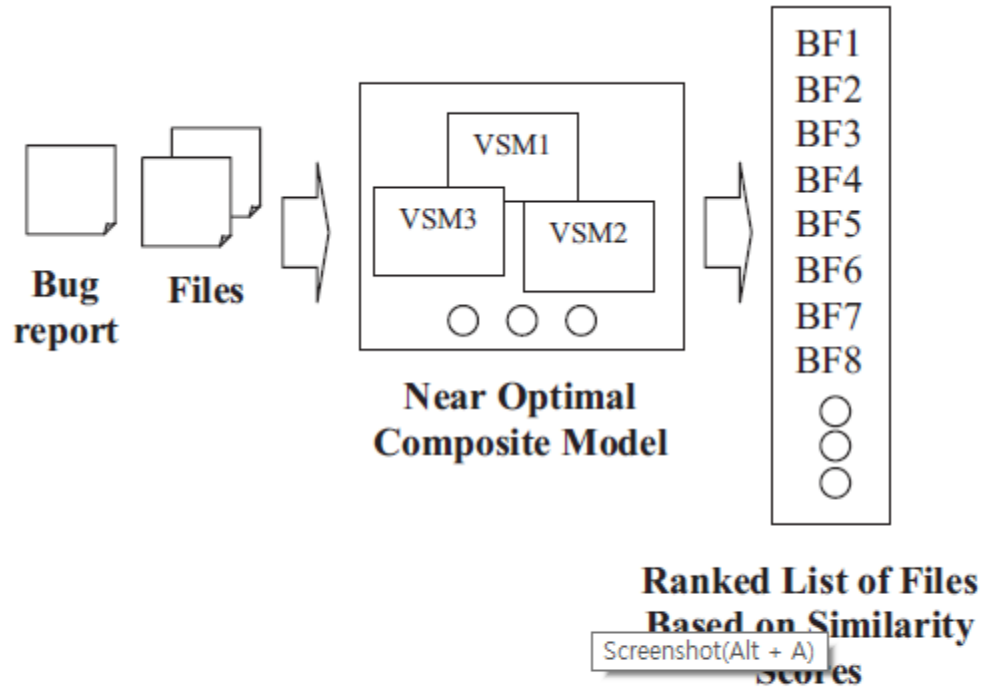


Fig. 2. Proposed Framework: Training and Deployment Phase

Objective Functions

- Search algorithms require an objective function to measure how good a candidate solution is. The goal of a genetic algorithm is to **maximize** the value of a given objective function.
- Before defining the objective function for GA, first introduce two evaluation **metrics** that are commonly used to measure the effectiveness of bug localization techniques:

  Mean Average Precision (MAP) 평균 정밀도

  Mean Reciprocal Rank (MRR) 평균 상호 순위

- **Mean Average Precision (MAP)**: MAP emphasizes all of the buggy files instead of only the first one. MAP is computed by taking the mean of the average precision scores across all bug reports.

$$AP = \sum_{k=1}^{M} \frac{P(k) \times pos(k)}{\#buggy\ files},$$

- **Mean Reciprocal Rank (MRR):** The reciprocal rank for a bug report is the reciprocal of the position of the first buggy file in the returned ranked files. MRR is the mean of the reciprocal ranks over a set of bug reports Q

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i}$$

- **Objective Function**: Our goal is to maximize MAP and MRR, as the higher their values are, the more effective the composite model is. Thus we define our objective function as:

$$\mathbf{obj = MAP + MRR}$$

Note:

1. Equal weight to MAP and MRR(1:1);
2. MAP is important if a developer is **interested to find all buggy files** by reading the recommended files one by one.
3. MRR is important if a developer is interested in only finding the first buggy file; based on this file, he can then **start his manual debugging effort** to find the other related buggy files.

5-A. AmaLgam

- AmaLgam which is a state-of-art **bug localization approach** incorporating three components to localize bugs in systems: version history, structure, and similar bug reports:
  - **Version history component**: Input commit logs collected from the version control system and outputs a list of files with their suspiciousness scores. $score_H(b, f)$
  - **Structure component**: Input the source code corpus and a given bug report and returns a list of files with their suspiciousness scores. $score_S(b, f)$
  - **Similar report component:** considers historical bug reports that have already been fixed. $score_R(b, f, B)$

A-1. Version history component

Input commit logs collected from the version control system and outputs a list of files with their suspiciousness scores.

$$score_H(b, f) = \sum_{c \in R \wedge f \in c} \frac{1}{1 + e^{12(1-((k-t_c)/k))}}$$

1. R is the set of bug fixing commits that were made in the last k days before bug report b was submitted;
2. t_c is the number of days that have elapsed between a commit c and bug report b;
3. k =15.

A-2. Structure component

Input the source code corpus and a given bug report and returns a list of files with their suspiciousness scores.

$$score_S(b, f) = \sum_{fp \in f} \sum_{bp \in b} sim(fp, bp)$$

1. fp is a category from file f;
2. bp is a category from bug report b;
3. sim(fp, bp) is the cosine similarity of the vector representations of fp and bp;

A-3. Similar report component

Considers historical bug reports that have already been fixed.

$$score_R(b, f, B) = \sum_{b' \in B} I(f, b') \times \frac{sim(b, b')}{|b'.Fix|}$$

1. I(f, b) is an indicator function that returns 1 if f is among the set of files that were modified to fix bug report b and it returns 0 otherwise;
2. sim(b, b) is the similarity of bug report b and a historical fixed bug report b in B,;
3. b.Fix is the set of files that are modified to fix bug report b;
4. |b.Fix | is the size of the set b.Fix ;
5. sim(b, b) is calculated using the procedure described in [12].

5-B. Compositional Model: *AmaLgam*$_{composite}$

- Present one strategy for combining our compositional VSM with AmaLgam.
- We combine the VSM models with different tf-idf weighting schemes, and three components of AmaLgam as follows.
- Given a bug **report b** and a set of historical fixed bug **reports B**, we compute the suspiciousness score $M_{Composite}$ (b, f) of file f as follows:

$$\sum_{i=1}^{15} w_i \times VSM_i(b,f) + \sum_{J}^{J \in H,R,S} w_j \times score_J(b,f)$$

6.Empirical Evaluation

A. Experimental Setting

- 1) Datasets: We use three datasets containing a total of 3,459 bug reports from three popular **open source projects**, AspectJ, Eclipse, and SWT.

TABLE IV. DATASET DETAILS

Project	Description	Period	#Fixed Bugs	#Source Files
AspectJ	Aspect-oriented extension of Java	07/2002-10/2010	286	6485
Eclipse	Open source IDE	10/2004-03/2011	3075	12863
SWT	Open source widget toolkit	10/2004-04/2010	98	484

A. Experimental Setting

- 2) **Effectiveness Calculation**: We use the components of our objective function, MAP and MRR, to evaluate the effectiveness of our solution. We also use **Hit@N.**
- **Hit@N**: This metric calculates the number of bug reports where one of its buggy files appears in the top N ranked files. Given a bug report, if at least one of its relevant files is in the top N ranked files, we consider the report is successfully located.

- **VSM$_{natural}$**: VSM with the standard tf-idf weighting scheme
- **VSM$_{composite}$**: Standard tf-idf weighting scheme combining our compositional VSM

- **AmaLgam$_{composite}$**: Combined the 15 VSMs with the 3 components of AmaLgam
- **AmaLgam$_{natural}$**: Natural AmaLgam

TABLE V. PERFORMANCE COMPARISONS. $AmaL = AmaLgam$. $AmaL_{compo.} = AmaLgam_{composite}$.

Project	Approach	Hit@1	Hit@5	Hit@10	MAP	MRR
AspectJ	$VSM_{natural}$	25 (8.7%)	43 (15.0%)	65 (22.3%)	0.05	0.13
	$VSM_{compo.}$	33 (11.5%)	55 (19.2%)	67 (23.4%)	0.07	0.16
	$AmaL$	127 (44.4%)	187 (65.4%)	209 (73.1%)	0.33	0.54
	$AmaL_{compo.}$	**145 (50.7%)**	**211 (73.8%)**	**227 (79.4%)**	**0.43**	**0.61**
Eclipse	$VSM_{natural}$	116 (3.8%)	456 (14.8%)	709 (23.1%)	0.01	0.01
	$VSM_{compo.}$	116 (3.8%)	544 (17.7%)	845 (27.5%)	0.01	0.01
	$AmaL$	1060 (34.5%)	1775 (57.7%)	2059 (67.0%)	0.35	0.45
	$AmaL_{compo.}$	**1108 (36.1%)**	**1905 (62.0%)**	**2187 (71.2%)**	**0.39**	**0.48**
SWT	$VSM_{natural}$	12(50.7%)	37 (73.8%)	49 (79.4%)	0.21	0.24
	$VSM_{compo.}$	14 (50.7%)	40 (73.8%)	53 (79.4%)	0.23	0.26
	$AmaL$	61 (62.2%)	80 (81.6%)	**88 (89.8%)**	0.62	**0.71**
	$AmaL_{compo.}$	**62 (63.2%)**	**83 (82.6%)**	**88 (89.8%)**	**0.63**	**0.71**

Conclusion

- In this paper, we build a solution that combines 15 VSMs with different tf-idf weighting schemes into an improved **composite model**, constructed using a **genetic algorithm**.
- We have evaluated our approach on 3,459 bug reports from AspectJ, Eclipse, and SWT and demonstrate that our approach can achieve **better performance**. Compared with $VSM_{natural}$, averaging across the 3 datasets, our approach, VSMcomposite, **improves** $VSM_{natural}$ in terms of Hit@5, MAP, and MRR by 18.4%, 20.6%, and 10.5% respectively.
- We have also combined the 15 VSMs with the 3 components of AmaLgam, which is the state-of-the-art bug localization technique. Compared with AmaLgam, averaging across the 3 datasets, $AmaLgam_{composite}$ can **improve** AmaLgam in terms of Hit@5, MAP, and MRR by 8.0%, 14.4% and 6.5% respectively.

THANK YOU