Empirical Software Engineering (2021) 26: 14
https://doi.org/10.1007/s10664-020-09898-5

Automatically recommending components for issue reports using deep learning

Morakot Choetkiertikul[1] · Hoa Khanh Dam[2] · Truyen Tran[3] · Trang Pham[3] · Chaiyong Ragkhitwetsagul[1] · Aditya Ghose[2]

Accepted: 14 October 2020 / Published online: 2 February 2021
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC part of Springer Natu

Content

Section 2.

- issue-driven software development
- LSTM

Section 3.

- overall framework of DeepSoft-C

Section 4.

- . Feature learning and extraction

Section 5.
predictive model

Section 6.
evaluation of approaches

Section 7.
Related work

Section 8.
Future work

2 Issue-driven software development

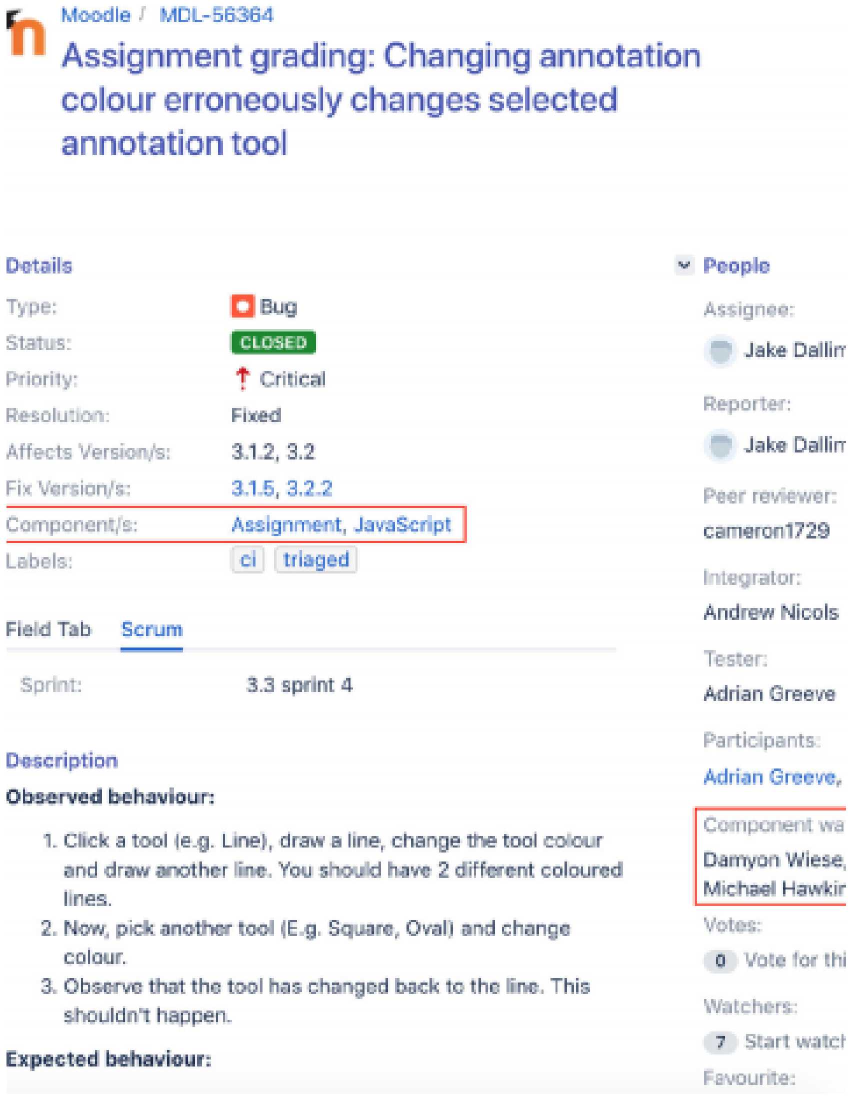
개발자에게 issue를 task로 할당하여 업무를 분할하는 방식

논문의 목표

- Deepsoft-C

(Deep learning model for Software Component recommendation)

- textual description of an issue
- predictive machinery
- recommends a list of components



1 Example of an issue in JIRA software

2 LSTM

발전된 RNN

Cell 대신 Memory block

- Forget gate
- Input gate
- Output gate

Feedforward
이전의 정보를 다음 예측에 사용한다.

확률적 경사 하강법

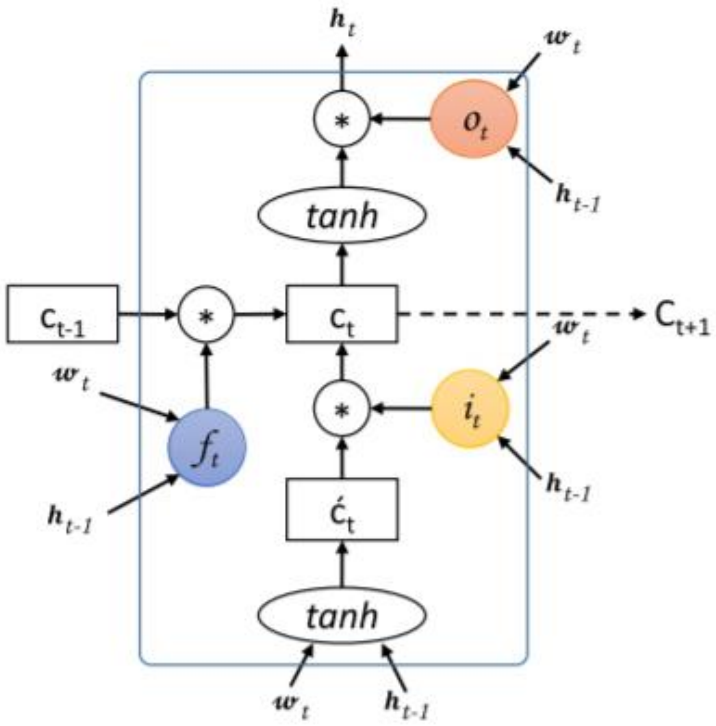


Fig. 3 The internal structure of an LSTM unit

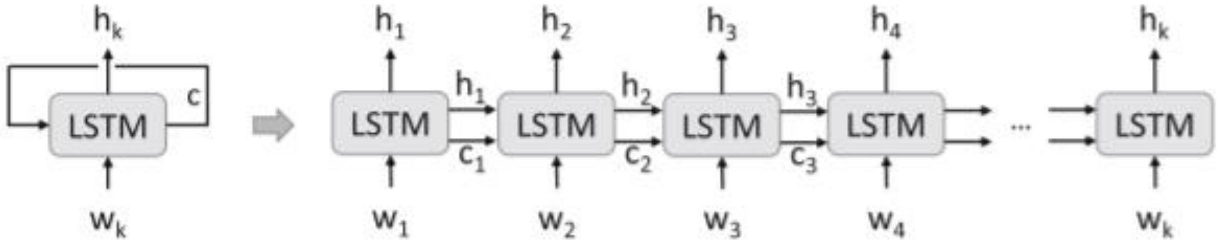


Fig. 2 An LSTM network

Approach Overview

1. LSTM으로 semantic feature 자동 추출
-> 딥러닝 머신이 특징 추출해줌

2. semantic feature를 기존의 textual silimarity feature와 결합

3. multi-label NN classifier
-> 학습과 추측 구성요소 간의 상관관계를 더 많이 포함가능

3 Model Architecture

LSTM을 통해

1) Semantic feature과

2) textual similarity를 학습한다.

두 feature를 combin하여 새로운 vector를 만든다.

=> Semantic feature + textual similarity

multi-label 분류 모델로 sinlge-layer NN을 사용해 component label 예측한다.

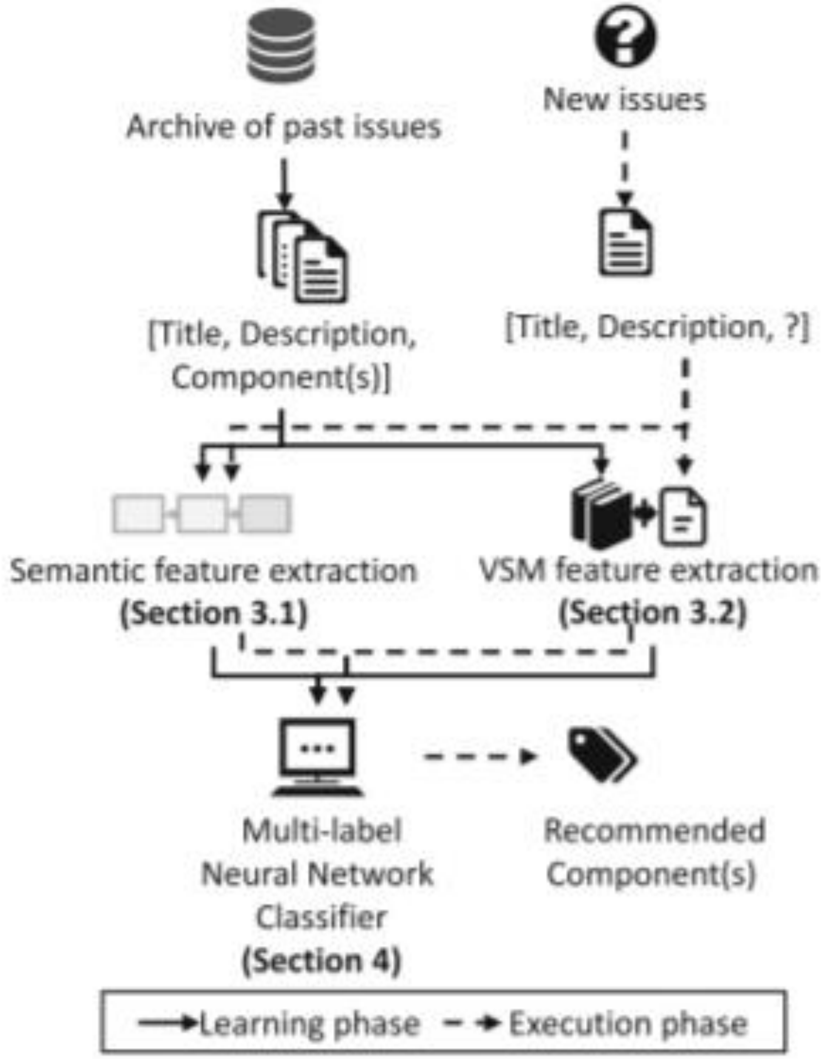


Fig. 5 The architecture of *DeepSoft-C*

4 Feature Learning and Extraction

4.1 Semantic Features

1. Issue를 단어로 parsing
2. LSTM 입력층: 파싱한 단어 워드 임베딩(vocabulary)
3. Output을 Mean pooling을 하여 sequence embedding 출력

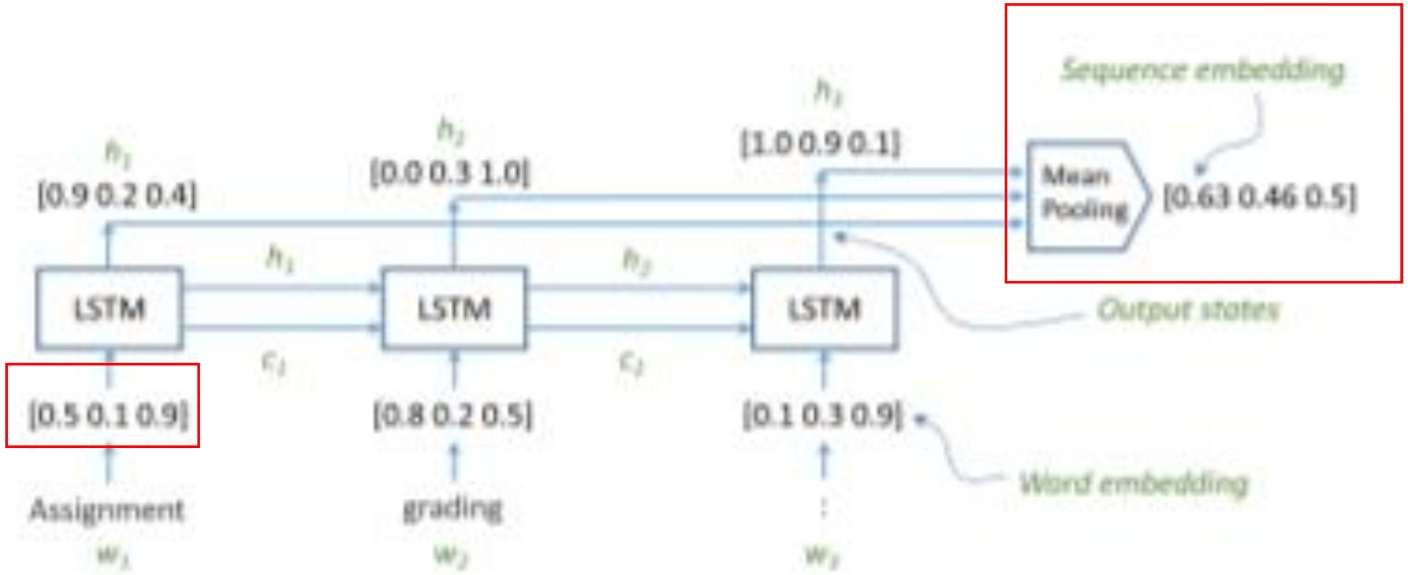


Fig. 6 An example of how a vector representation of an issue description, i.e., sequence embedding, is generated from a sequence of words in the description using LSTM

4 Feature Learning and Extraction

4.1.1 Learning Semantic Features

-LSTM model parameters and the word embedding matrix 학습

-Loss는 cross entropy loss 사용 : 발전된 MSE

$$P\left(w_{t+1}=k \mid w_{1:t}\right)=\frac{\exp\left(\mathcal{U}_k^{\top} \boldsymbol{h}_t\right)}{\sum_{k'} \exp\left(\mathcal{U}_{k'}^{\top} \boldsymbol{h}_t\right)}.$$

$$L(\mathcal{P})=-\log P(w_1)-\sum_{t=1}^{n-1} \log P\left(w_{t+1} \mid \boldsymbol{w}_{1:t}\right)$$

4.1.1 Learning Semantic Features

확률적 경사하강법: P(M, U, internal LSTM parameter) Update

Theano 라이브러리: LSTM 구현

실험적으로 Hyper-parameter(learning rate, batch size) 결정

매 epoch 종료시: Validation set 사용

Overfitting 예방: dropout- regularization 기법

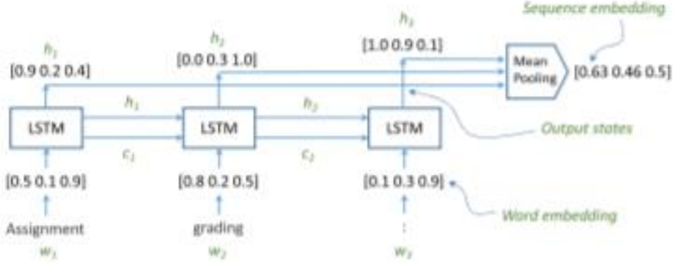


Fig. 6 An example of how a vector representation of an issue description, i.e., sequence embedding, is generated from a sequence of words in the description using LSTM

final semantic feature vector: : Mean pooling

NoiseContrastive Estimation 샘플링 기법: computational time 감소

성능 계산 지표: PPL 낮추기

4.2 Textual Similarity Features

1) Component Label에 따라 모든 issue를 합친 description 구성

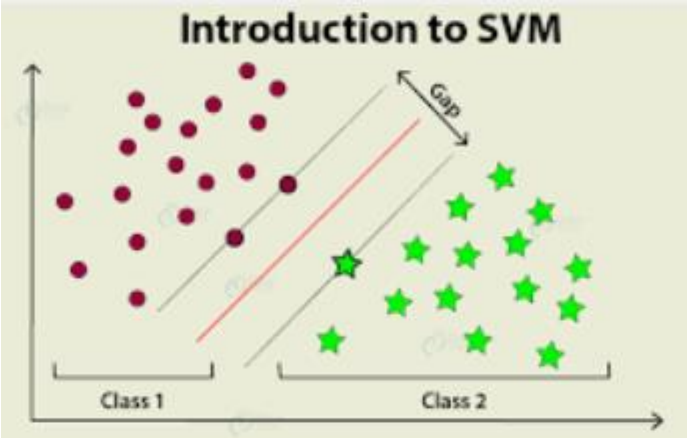
$$Similarity(C_i, S_j) = \frac{\mathbf{V}_{C_i} \bullet \mathbf{V}_{S_j}}{|\mathbf{V}_{C_i}|\,|\mathbf{V}_{S_j}|}$$

** C는 component, S는 issue report

2) Frequency 계산

$$tf(t, desc) = \frac{f_{t,desc}}{T} \text{ and } idf(t) = \log\left(\frac{D}{n_t}\right)$$

5 Predictive Model



Traditional Classifier

SVM, Random Forests

Multi-labeling

일반적인 방법: Binary Relevance

단점: component 간 correlation 정보 부족

(JavaScript, GUI)

대안: Simple Neural Network

5 Predictive Model

Hyper Parameters

500 epochs with batch size 100.
Learning rate = 0.02
Adaptation rate = 0.9
Smoothing factor 1-^-6

입력층

semantics + textual similartiy features

은닉층

ReLU : activation function
Sigmoid: Overfitting 예방

출력층

확률이 가장 높은 k개의 component

$$f_\Theta(\mathbf{x}) = sigmoid(\mathbf{W}_o relu(\mathbf{W}_h x + \mathbf{b}_h) + \mathbf{b}_o)$$

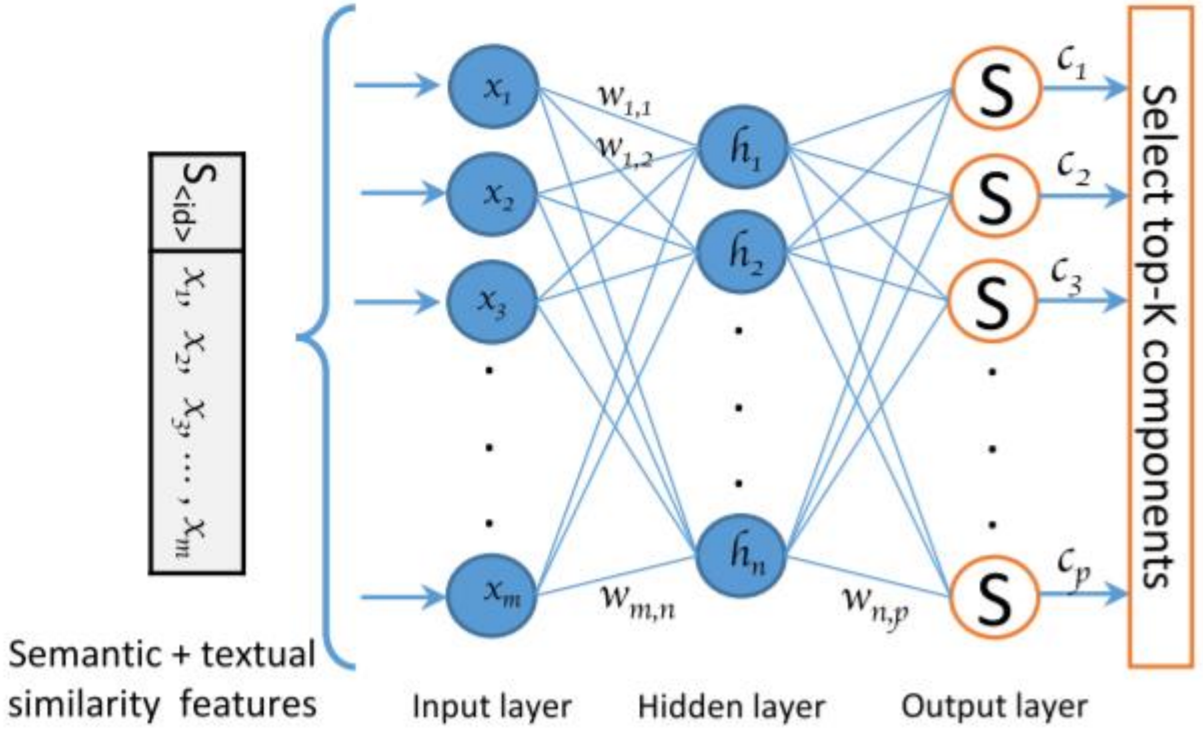


Fig. 7 A multi-labelling neural network classifier

6
Evaluation

- 데이터셋과 개발 프로세스 과정
- 실험적 세팅
- 퍼포먼스 측정
- 결과 리포트

6.1 Research Questions

RQ 1: Sanity Check

RQ2: Compare to the Other Techniques

-> Deepsoft-C

비교 대상: LDA-KL, TDF-IF

RQ3: Benefits of LSTM (feature 학습)

Modeling: CNN

비교 대상: 구글의 Doc2Vec

6.1 Research Questions

RQ4: : Benefits of Combining LSTM and Textual

비교 대상: LSTM만 썼을 때,

textual similarity feature만 썼을 때

RQ5: Benefits of the Multi-label Classifier

비교 대상: SVM, Random Forest (binary classifier)

Datasets

- 11개의 오픈소스 프로젝트:
- Apache, Duraspace, Atlassian, Moodle

- Filtering step
- 209218개 Closed, resolved 상태, fixed된 상태 의 이슈
- 142025개, 829 components
- 몇몇 Non ASCII 문자 제거,

최종적으로,
50,000 issues from a repository

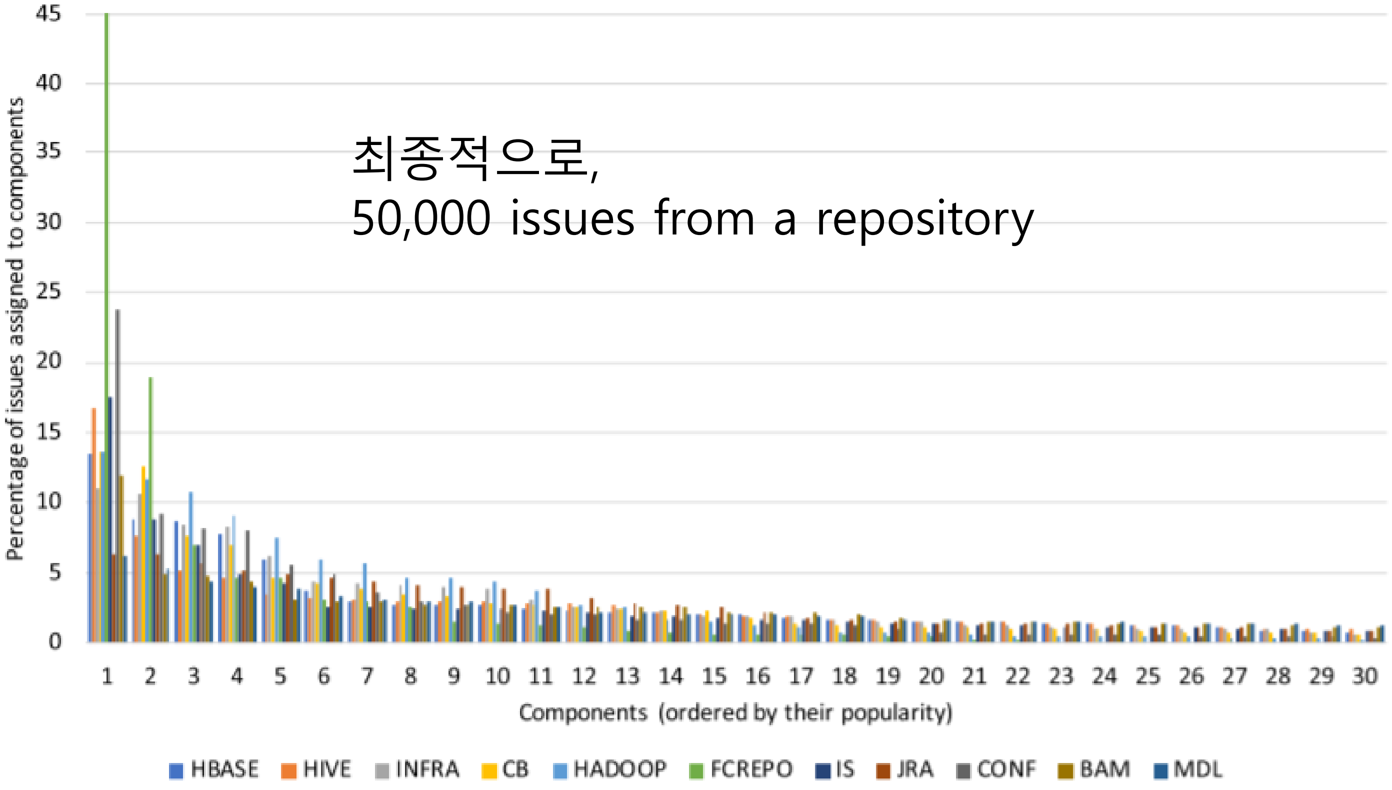


Fig. 8 The distribution of issues in the 30 most popular components in each project

Experimental Settings

- issue creation time기준 dataset 분류 (한계점)
  Training / Validation / Test set – 60:20:20
  시간별로 training < validation < test 최신

- 각 프로젝트 별로 따로 학습을 수행
  - 프로젝트마다 컴포넌트의 종류가 다름

- validation set을 사용
  매 epoch마다 hyper-parmeter를 조정
  model parameter값은 training data에만 영향

Performance Measures

Recall@k : k개의 추천된 구성요소 결과에서 올바른 결과를 반영합니다.

$$Recall@k = \frac{1}{m}\sum_{i=1}^{m}\frac{|Rec_i \cap Label_i|}{|Label_i|}$$

Axy statistic :

$$\hat{A}_{XY} = \frac{[\#(X > Y) + (0.5 \times \#(X = Y))]}{n},$$

결과 소개

Outperforms

- Most Popular (MP) method (baseline)
- two state-of-the-art benchmarks
- six alternative techniques

Results

- 추천하는 컴포넌트 수, 즉 k가 작을수록 올바른 추천을 한다.
- the highest class imbalance situation

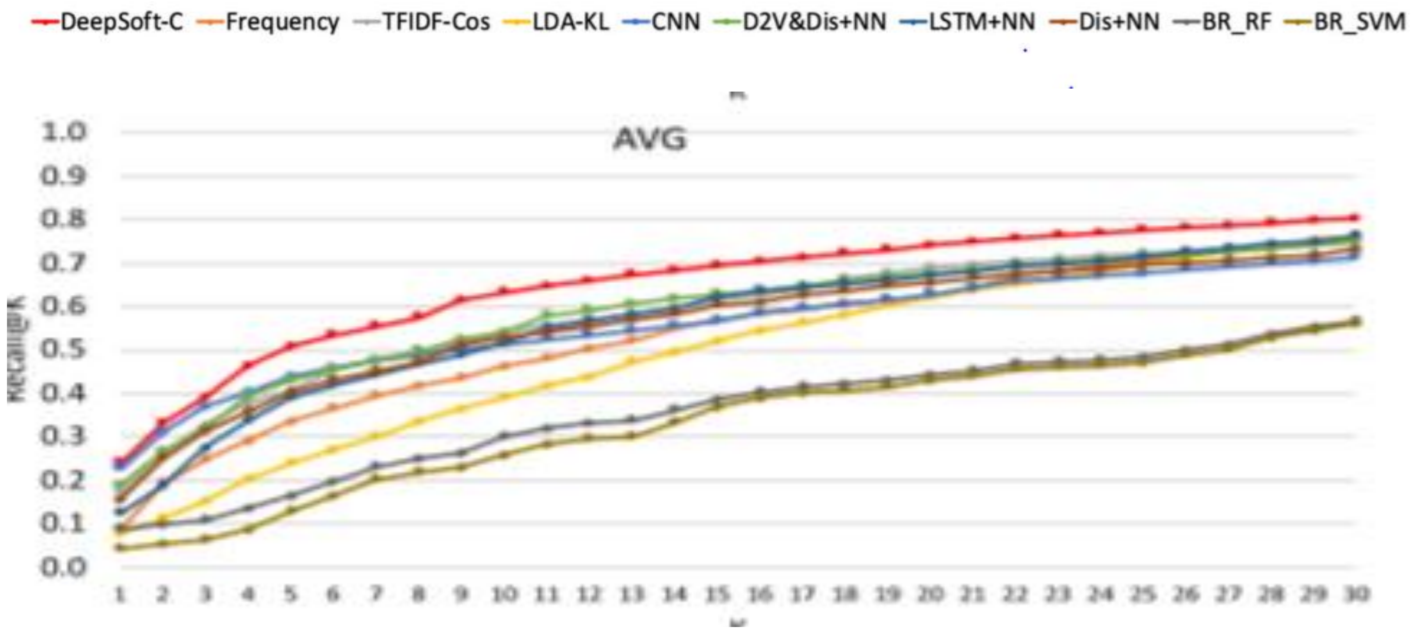


Fig. 9 The performance of DeepSoft-C and the other methods

RQ1: Sanity Check

- Deepsoft-C outperforms the baseline benchmark using MP.
- K의 수가 작을 수록 더 나은 performance.

Table 3 Evaluation results of DeepSoft-C and the most popular method (frequency)

Project	Method	Recall@k					
		k = 5	%imp	k = 10	%imp	k = 15	%imp
Average	DeepSoft-C	0.508		0.631		0.694	
	Frequency	0.336	51.26	0.461	36.88	0.569	22.10

RQ2: Compare to the Other Techniques

- Component recommendation 분야의 기법과 비교

Table 4 Evaluation results of DeepSoft-C, TFIF-Cos, and LDA-KL

Project	Method	Recall@k					
		k = 5	%imp	k = 10	%imp	k = 15	%imp
	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
FCREPO	DeepSoft-C	0.522		0.774		0.845	
	CNN	0.496	5.22	0.532	45.43	0.625	35.27
	D2V&Dis+NN	0.408	27.94	0.555	39.46	0.880	−3.98
Average	DeepSoft-C	0.508		0.631		0.694	
	TFIDF-Cos	0.406	25.15	0.540	16.98	0.616	12.64
	LDA-KL	0.240	111.48	0.390	61.83	0.520	33.43

The percentage (%imp) of the improvement brought by DeepSoft-C over the other methods computed from the mean of Recall@5, 10, and 15

RQ3: Benefits of LSTM

- CNN과 Doc2Vec에 비해 좋은 성능
- 역시 k에 따라 성능이 다르지만, LSTM은 이러한 영향을 완화.

Table 5 Evaluation results of DeepSoft-C, CNN, and D2V&Dis+NN

Project	Method	Recall@k					
		k = 5	%imp	k = 10	%imp	k = 15	%imp
Average	DeepSoft-C	0.508		0.631		0.694	
	CNN	0.437	16.05	0.513	23.01	0.566	22.69
	D2V&Dis+NN	0.429	18.41	0.540	16.84	0.628	10.46

RQ4: Benefits of Combining LSTM and Textual Similarity Features

- LSTM+NN: semantic features
- Dis+NN: textual similarity features
- 성능이 좋아졌다 -> 두 feature는 서로 보완 효과가 있다

Table 6 Evaluation results of DeepSoft-C, LSTM+NN, and Dis+NN

Project	Method	Recall@k					
		k = 5	%imp	k = 10	%imp	k = 15	%imp
Average	DeepSoft-C	0.508		0.631		0.694	
	LSTM+NN	0.391	29.96	0.518	21.97	0.622	11.62
	Dis+NN	0.404	25.69	0.527	19.77	0.604	15.02

RQ5: Benefits of the Multi-label Classifier

- Compare to Binary Relevance, using SVM, RF (Sci-kit learn)
- 큰 성능 향상

Table 7 Evaluation results of DeepSoft-C, BR_RF, and BR_SVM

Project	Method	Recall@k					
		k = 5	%imp	k = 10	%imp	k = 15	%imp
Average	DeepSoft-C	0.508		0.631		0.694	
	BR_RF	0.172	194.46	0.329	91.82	0.402	72.53
	BR_SVM	0.138	266.82	0.297	112.75	0.387	79.50

결론

Issue textual description에 component에 대한 언급과 상관없이 추천가능.

Moodle / MDL-52768

Feedback not affected by marking workflow in gradebook

Details

Type:	Bug	Status:	CLOSED
Priority:	Major	Resolution:	Not a bug
Affects Version/s:	2.8.10, 3.0	Fix Version/s:	None
Component/s:	Assignment		
Labels:	triaged		
Affected Branches:	MOODLE_28_STABLE, MOODLE_30_STABLE		

Description

Marking Workflow does not work for projects restored from a full course backup.

We run courses over an academic year, and then backup/restore these (without student data) into the new academic year in early September. Last year we created projects with Marking Workflow which worked very successfully. However, this year, running the same projects in the duplicated courses, learners are able to view their feedback before the Workflow status is set to Released. This happens even though the project says Marking Workflow is operating.

I cannot confirm if this is the case in 2.6 but we are on 2.8, and I have tested also in 3.0 with the same

People

Assignee:

Unassigned

Reporter:

Phineas Head

Participants:

Mark Nelson, Phineas Head

Component watchers:

Adrian Greeve, Jake Dallimore, ... (4)

Favourite:

Assignment

결론

다른 이슈에 비해 Issue textual description이 짧으면 assignment라 추측 못했음.

Moodle / MDL-55163

Fix mod_assign_copy_previous_attempts

Export

Details

Type:	Bug
Priority:	Minor
Affects Version/s:	3.1
Component/s:	Assignment
Labels:	ci triaged
Testing Instructions:	1. Go to Administration, Web Services, API Documentation and check that mod_assign_copy_previous_attempt is displayed in the list of available functions
Affected Branches:	MOODLE_31_STABLE
Fixed Branches:	MOODLE_30_STABLE, MOODLE_31_STABLE
Pull from Repository:	git://github.com/jleyva/moodle.git
Pull Master Branch:	~~MDL-55163~~-master Core no status
Pull Master Diff URL:	https://github.com/jleyva/moodle/compare/5a1728d...MDL-55163-master
Documentation link:	https://docs.moodle.org/dev/Web_service_API_functions

Status:	CLOSED
Resolution:	Fixed
Fix Version/s:	3.0.6, 3.1.2

Description

The function is:

- Not exposed via WS
- Have a bug inside

People

Assignee:	Juan Leyva
Reporter:	Juan Leyva
Peer reviewer:	Jun Pataleta
Integrator:	David Monllaó
Tester:	Andrew Lyons
Participants:	Andrew Lyons, CiBoT, ...
Component watchers:	Adrian Greeve, ... (5)
Votes:	0 Vote for this issue
Watchers:	4 Start watching this issue
Favourite:	Add to favourite issu

Dates

Created:	09/Jul/16 4:13 AM
Updated:	22/Nov/16 3:59 AM
Resolved:	22/Jul/16 3:56 AM
Fix Release Date:	12/Sep/16

결론

- LSTM을 issue component 문제로 확장함.
- Issue report의 nature: many SW project terminologies

1) LSTM을 사용한 pre-trained word embedding
2) 두 feature의 결합

   Tradeoff: large embedding size -> overfitting & computational time

3) Long tail situation -> multi-labeling

   End to end와 달리, issue report를 나타내는 different set of feature를 사용하여 훈련 시간을 단축

7 Related Work

which software components should an issue be assigned to?

1) Text Classification

2) Issue Classification

8 Future Work

- 상업화 = 자본주의!
- 큰 규모의 프로젝트에 제안된 방식을 평가해보려 함.

한계점

- New component에 대한 예측이 불가능하다는 점

-> dataset을 issue creation 시간으로 나눔으로써

논문을 읽으며 느낀 점

위키백과 의외로 좋았다

감히 평가하자면 자연어처리의 LSTM의 과정을 서술한 구성으로, 전체적인 숲을 그리는데 도움이 되었습니다.

발표를 들어 주셔서
감사합니다