Auto-labelling of GitHub Issues using Natural Language Processing

Group 17: Larry Law (e0325467@u.nus.edu), Liu Zechu (e0323879@u.nus.edu), Ng Tek In (e0175456@u.nus.edu), Zhuang Xinjie (e0202855@u.nus.edu)

Motivations

[1] Popular GitHub Repositories have **thousands of open issues**.

[2] **Regex-based labellers tend not to generalise well** yet requiring **engineering effort** to design them.

[3] Issue-Label-Bot offered a plug-and-play **NLP model** but it uses a **simple GRU-based model architecture**, with **no special treatment of machine information**

Approach	Generalise well?	Minimal Engineering?	Accounts for Machine Info?
Ours	Yes	Yes	Yes
Regex	No	No	No
Issue-Label-Bot	Yes	Yes	No

Fig 2.1. Comparison of different approaches to automatically labelling GitHub issues.

Research Questions

[1] How effective is SOTA NLP-based approach relative to traditional regex approach in auto-labelling GitHub issues?

[2] How should the NLP-based labeller handle machine information?

Experiment Setup

Train Set: 80% of the issues in the Tensorflow, Rust, Kubernetes repositories ("seen repositories").
Test Set: 1) 20% of the issues in the seen repositories and 2), all issues in Flutter, OhMyZsh, and Electron ("unseen repositories")
Labels: Feature, Bug, Documentation
Baseline: Regex-based classifier
Metrics: Accuracy and Weighted F1 score

[1] Train and Test Sets have similar Distribution

	Feature	Bug	Docs
Train Set	0.3139	0.6164	0.06971
Test Set	0.3944	0.5267	0.07900

Fig 4.1.1 Statistics of issues in train (Tensorflow, Rust, Kubernetes) and test sets (Flutter, OhMyZsh, and Electron).

[2] % of Machine Information is significant

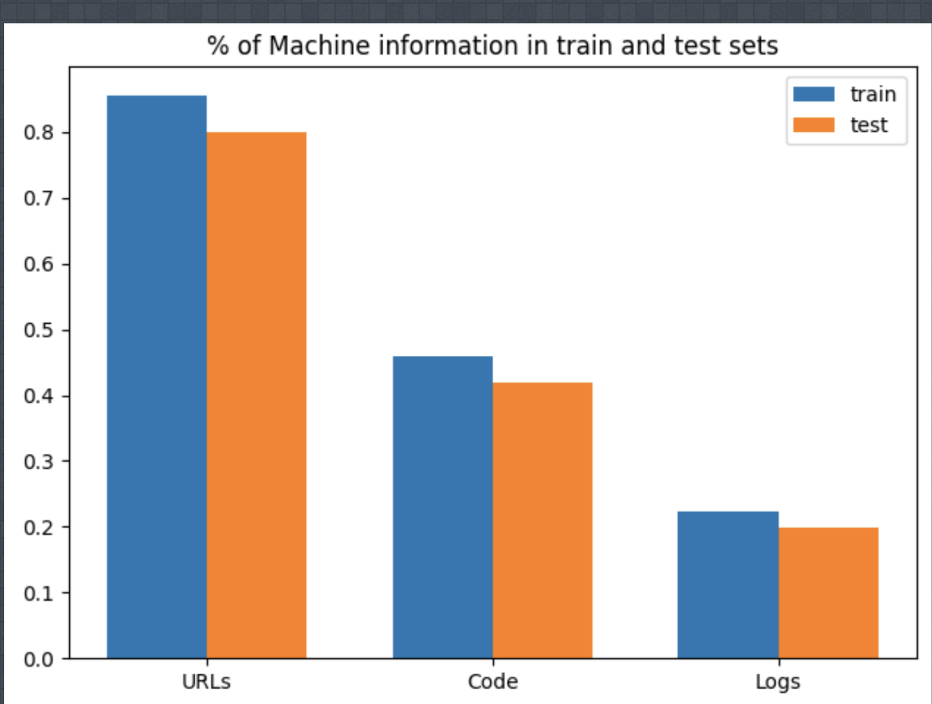

Fig 4.1.2: % of machine information (URL, Code, Logs) in train and test sets.

Methods

Feature-Based Classifier

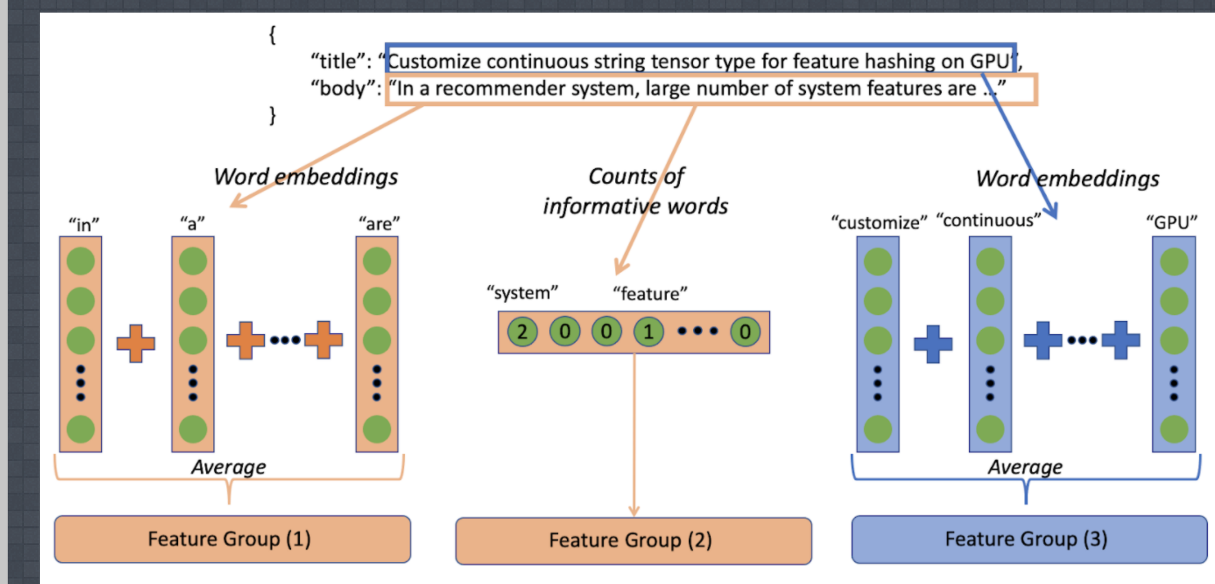

Fig 3.1: Feature groups used in feature-based classifier

Our engineered features include...

[1] For each issue, we **averaged normalised word embeddings of all tokens in the issue's body text.** These word embeddings were trained on training corpus using Gensim's **Word2Vec CBOW** algorithm.

[2] We **shortlisted a vocabulary set which comprises N most frequent words from each class** (bug/feature/doc), **excluding M most frequent words in the combined corpus**, in order to capture words important in distinguishing different categories. Then, for each issue, we generated a **word count vector using this shortlisted vocabulary** (N = 150, M = 50).

[3] Similar to [1], but applied to the **title** of the issue.

Combinations of these features are then passed into **Logistic Regression** and **Neural Network**.

Fine-Tuned BERT Sequence Classifier

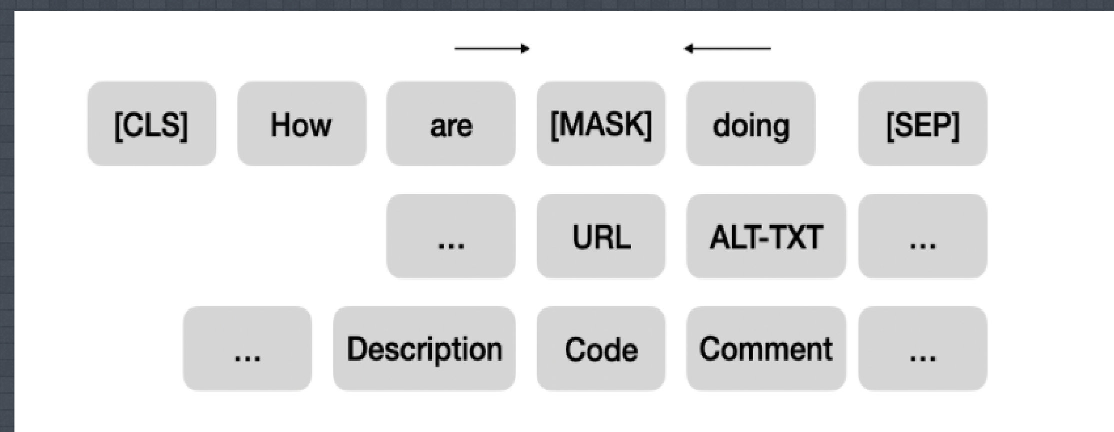

Fig 3.3: BERT's Masked Language Modeling pre training objective (first row) and our task of GitHub issue labelling (second and third row). In both tasks, bidirectional contextual information can benefit the model.

By pretraining on the "Masked-Language-Modelling" objective, BERT offers **deep bidirectional representations** that are jointly conditioned on both the left and right context in all layers. We used BERT as:

[1] BERT's pretrained **contextualised embeddings** contrast with the Feature-Based classifier's **static word embeddings**. Moreover, as **descriptions of machine information can come both before and after the information**, conditioning on context from both directions can be important.

[2] By classifying issues based on the aggregate sequence representation, **BERT may discover useful information within the sequence that our hand-designed features missed out.**

[3] BERT's fine-tuning based representation achieved **SOTA results on eleven NLP tasks.**

Filtering out natural-language-like machine information

[1] **Machine information**, which refers to text such as URLs, code blocks and logs etc, **can contain natural language vocabulary** which may not have the same semantic meaning.

[2] Given that such instances may interfere with the model, we decided to **filter out machine information** and examine whether our model will benefit from such exclusions.

[URL]	[lock-free persistent b+ tree](https://github.com/)
[Code block]	```\r\nlet mut mutable = 12;\r\nmutable = 21;\r\n// This line of syntax compilation error\r\nmutable = true;\r\n```
[Log]	FATAL: ThreadSanitizer: failed to intercept pthread_mutex_lock\r\n

Discussion & Ablation Studies

Claim 1: NLP approach is significantly more effective than the regex approach in auto-labelling GitHub issues.

	Seen Repos		Unseen Repos	
Approach	F1	Acc	F1	Acc
Feature-Based	0.8504	0.8529	0.8356	0.8362
BERT	0.9001	**0.9005**	0.8723	**0.8752**
Regex	0.4815	0.6256	0.3634	0.5267

Fig 4.3.1: Best results of each classifier when forced to make a classification on the test sets of seen and unseen repositories.

	Seen Repos		Unseen Repos	
Approach	F1	Acc	F1	Acc
Feature-Based	0.8458	0.8714	0.8243	0.8656
BERT	**0.8958**	0.9170	**0.8644**	0.9050
Regex	0.4815	0.6256	0.3634	0.5267

Fig 4.3.2 Best results of each classifier when not forced to label all issues.

Claim 2: Removing machine information led to better generalisation.

	Seen Repos		Unseen Repos	
Approach	Avg %Δ in F1 (10^-3)	Avg %Δ in acc (10^-3)	Avg %Δ in F1 (10^-3)	Avg %Δ in acc (10^-3)
Filter Code	-2.126	-1.341	2.475	2.562
Filter URL	-2.191	-2.300	6.080	5.944
Filter Log	-0.1041	-0.5185	2.304	0.2166

Fig 4.3.3: Results of how filtering out each machine information affected BERT's performance on the test examples with the information. %Δ := (performance from filtering X on test sets with X - original performance) / original performance, where X is either code, url, or logs.

Claim 3: Natural Language Information matters more than Machine Information

	Seen Repos		Unseen Repos	
Approach	F1	Acc	F1	Acc
All	**0.8997**	**0.9003**	0.8750	0.8784
All w/o Title	0.8689	0.8700	0.8234	0.8301
All w/o Body	0.8445	0.8455	0.8379	0.8407
All w/o Filtering Code	0.8982	0.8986	0.8738	0.8760
All w/o Filtering URL	0.8986	0.8992	0.8741	0.8772
All w/o Filtering Log	0.8975	0.8981	**0.8758**	**0.8785**

Fig 5.1: Ablation study on the BERT model trained on both title and body text, wherein code, URL, and logs machine information have been filtered out.

References

M. K. (n.d.). *NUS CS4248 Natural Language Processing*. Lecture.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, & Kristina Toutanova. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

Tomas Mikolov, Kai Chen, Greg Corrado, & Jeffrey Dean. (2013). Efficient Estimation of Word Representations in Vector Space.