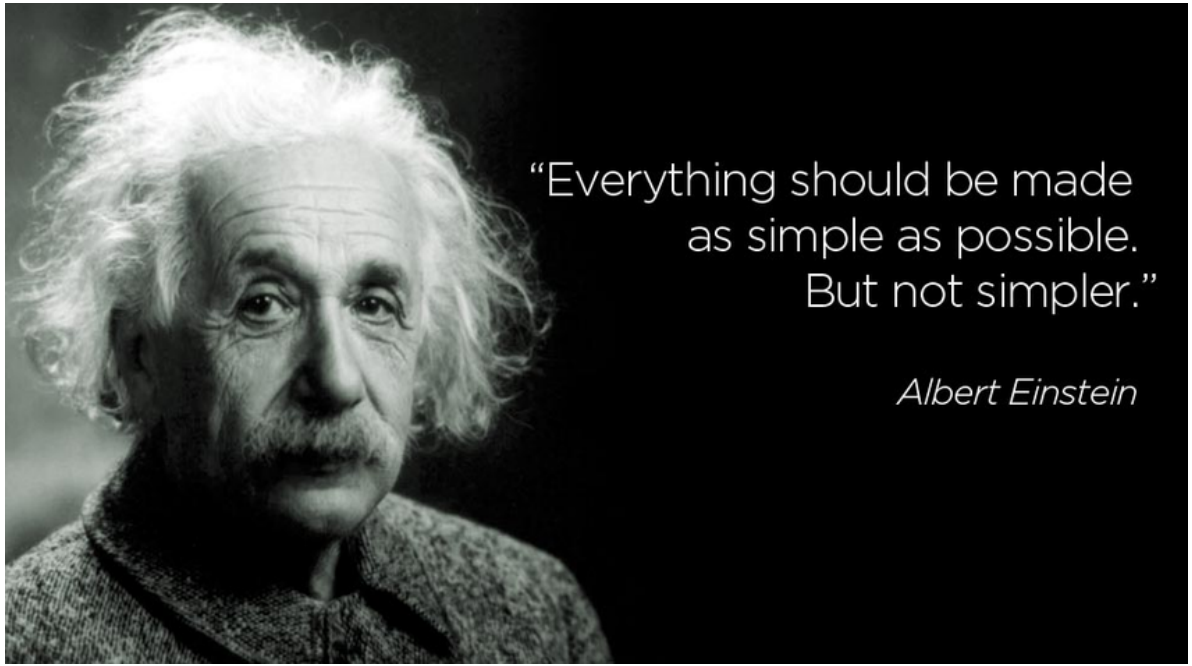
Prediction of Relatedness in Stack Overflow: Deep Learning vs. SVM

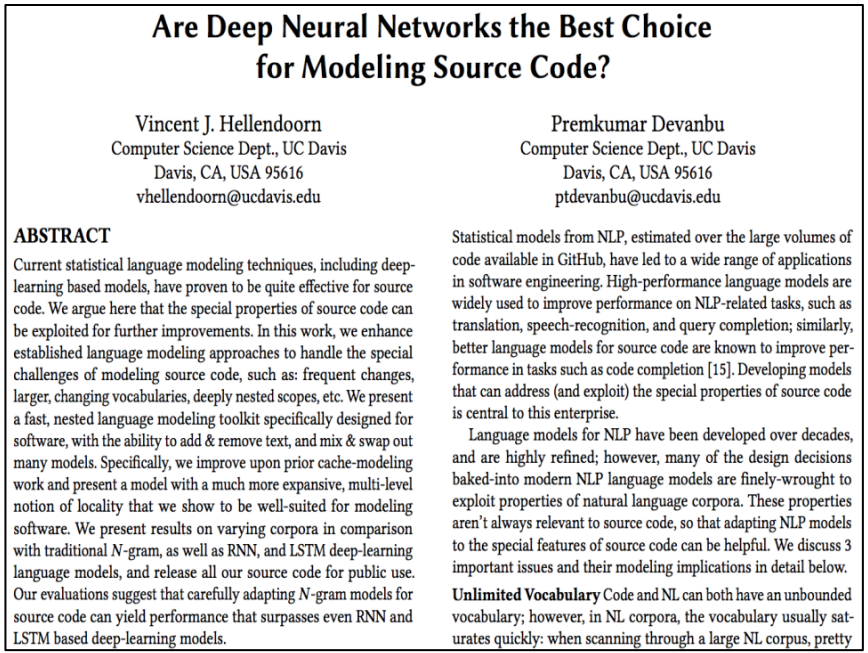
A Reproducibility Study

Bowen Xu, Amirreza Shirani, David Lo, Mohammad Amin Alipour

“Try-with-simple” Practice



“Try-with-simple” Practice



Are Deep Neural Networks the Best Choice for Modeling Source Code?

Vincent J. Hellendoorn
Computer Science Dept., UC Davis
Davis, CA, USA 95616
vhellendoorn@ucdavis.edu

Premkumar Devanbu
Computer Science Dept., UC Davis
Davis, CA, USA 95616
ptdevanbu@ucdavis.edu

ABSTRACT

Current statistical language modeling techniques, including deep-learning based models, have proven to be quite effective for source code. We argue here that the special properties of source code can be exploited for further improvements. In this work, we enhance established language modeling approaches to handle the special challenges of modeling source code, such as: frequent changes, larger, changing vocabularies, deeply nested scopes, etc. We present a fast, nested language modeling toolkit specifically designed for software, with the ability to add & remove text, and mix & swap out many models. Specifically, we improve upon prior cache-modeling work and present a model with a much more expansive, multi-level notion of locality that we show to be well-suited for modeling software. We present results on varying corpora in comparison with traditional *N*-gram, as well as RNN, and LSTM deep-learning language models, and release all our source code for public use. Our evaluations suggest that carefully adapting *N*-gram models for source code can yield performance that surpasses even RNN and LSTM based deep-learning models.

Statistical models from NLP, estimated over the large volumes of code available in GitHub, have led to a wide range of applications in software engineering. High-performance language models are widely used to improve performance on NLP-related tasks, such as translation, speech-recognition, and query completion; similarly, better language models for source code are known to improve performance in tasks such as code completion [15]. Developing models that can address (and exploit) the special properties of source code is central to this enterprise.

Language models for NLP have been developed over decades, and are highly refined; however, many of the design decisions baked-into modern NLP language models are finely-wrought to exploit properties of natural language corpora. These properties aren't always relevant to source code, so that adapting NLP models to the special features of source code can be helpful. We discuss 3 important issues and their modeling implications in detail below.

Unlimited Vocabulary Code and NL can both have an unbounded vocabulary; however, in NL corpora, the vocabulary usually saturates quickly: when scanning through a large NL corpus, pretty

[FSE’17, Vincent J. Hellendoorn and Premkumar Devanbu]

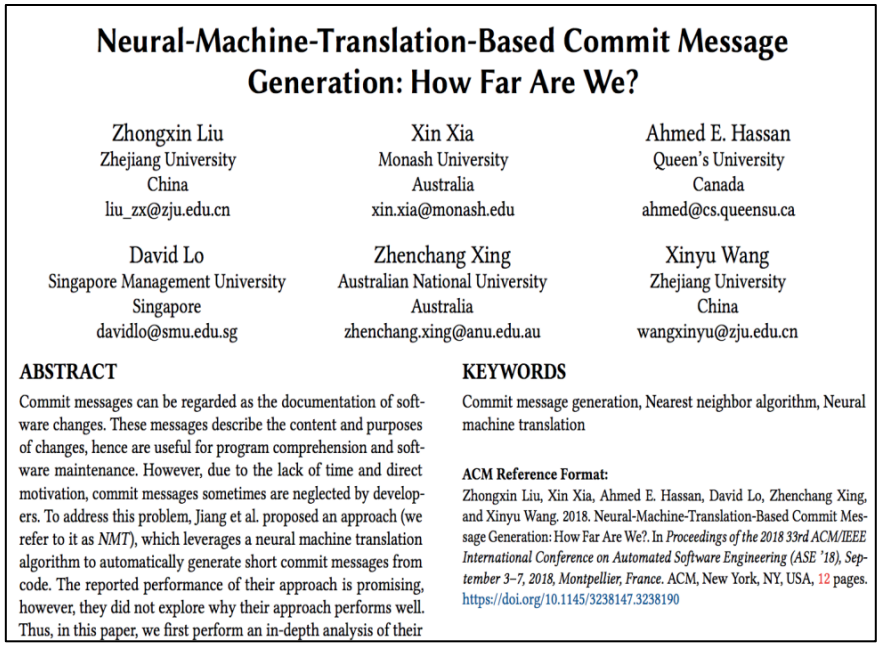
Deep Learning (RNN) vs. N-gram

“

Our evaluations suggest that carefully adapting N-gram models for source code can yield performance that **surpasses** even RNN and LSTM based **deep-learning models**.

”

“Try-with-simple” Practice



Neural-Machine-Translation-Based Commit Message Generation: How Far Are We?

Zhongxin Liu
Zhejiang University
China
liu_zx@zju.edu.cn

Xin Xia
Monash University
Australia
xin.xia@monash.edu

Ahmed E. Hassan
Queen’s University
Canada
ahmed@cs.queensu.ca

David Lo
Singapore Management University
Singapore
davidlo@smu.edu.sg

Zhenchang Xing
Australian National University
Australia
zhenchang.xing@anu.edu.au

Xinyu Wang
Zhejiang University
China
wangxinyu@zju.edu.cn

ABSTRACT

Commit messages can be regarded as the documentation of software changes. These messages describe the content and purposes of changes, hence are useful for program comprehension and software maintenance. However, due to the lack of time and direct motivation, commit messages sometimes are neglected by developers. To address this problem, Jiang et al. proposed an approach (we refer to it as *NMT*), which leverages a neural machine translation algorithm to automatically generate short commit messages from code. The reported performance of their approach is promising, however, they did not explore why their approach performs well. Thus, in this paper, we first perform an in-depth analysis of their

KEYWORDS

Commit message generation, Nearest neighbor algorithm, Neural machine translation

ACM Reference Format:

Zhongxin Liu, Xin Xia, Ahmed E. Hassan, David Lo, Zhenchang Xing, and Xinyu Wang. 2018. Neural-Machine-Translation-Based Commit Message Generation: How Far Are We?. In *Proceedings of the 2018 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE ’18), September 3–7, 2018, Montpellier, France.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3238147.3238190

[ASE’18, Liu et al., Distinguish Paper]

Deep Learning (RNN) vs. KNN

“Our study shows that it is worth **trying simple and fast methods** before applying complicated and time-consuming techniques on software engineering tasks.”

Does “Try-with-simple” practice still hold on different sizes of datasets?

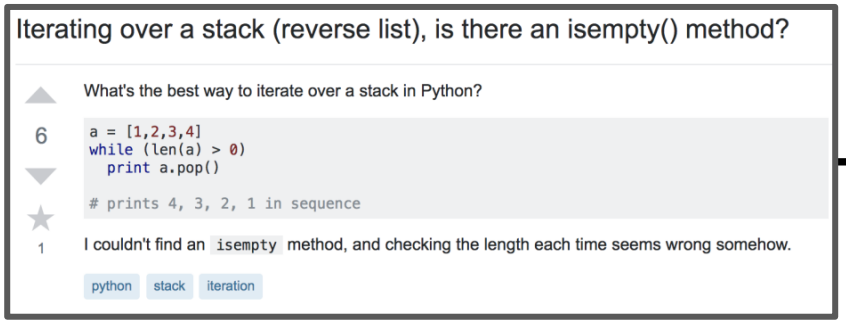
A Reproducibility Study on Prediction of Relatedness in Stack Overflow

1. Build a larger dataset.
2. Reproduce state-of-the-art approaches.
3. Introduce a simpler and faster approach.

Relatedness in Stack Overflow

Question A

Question B



Duplicate

Direct

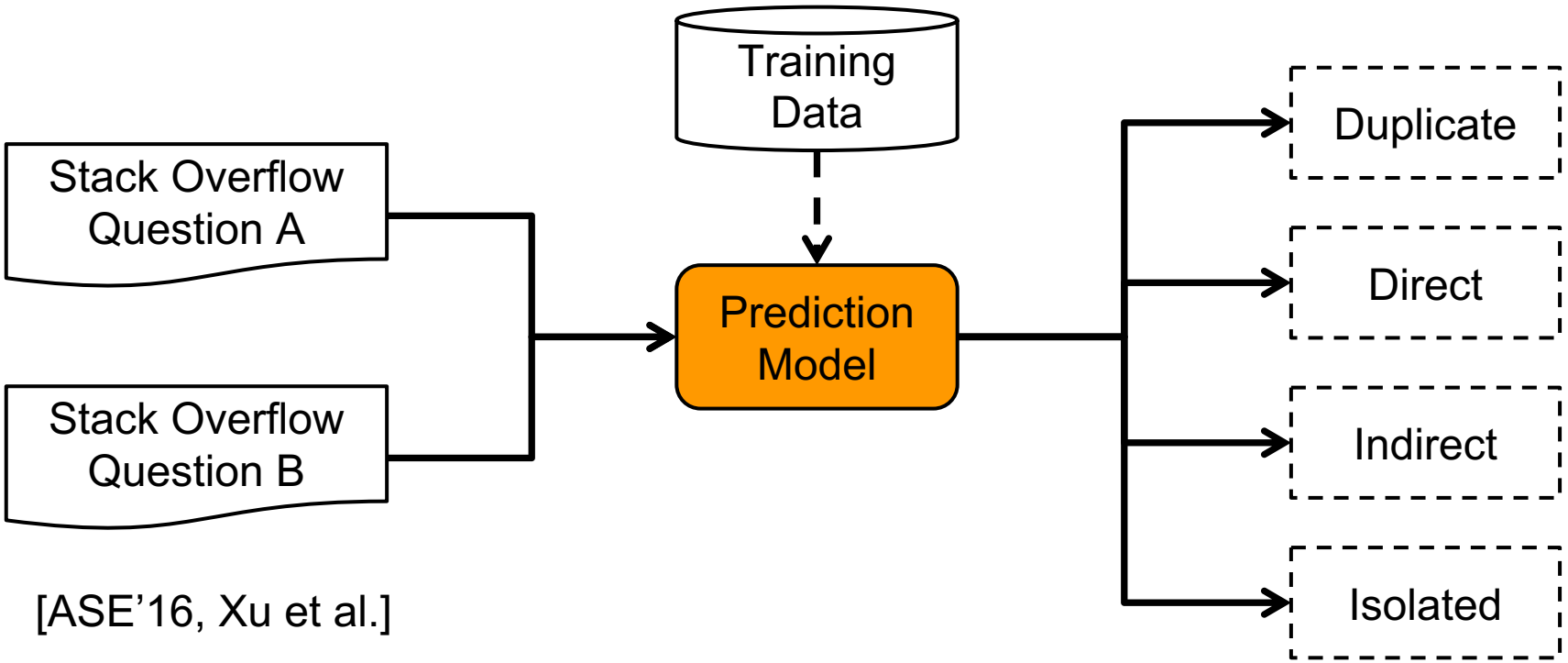
Indirect

Isolated

[ASE'16, Xu et al.]

Multiclass relatedness would support more targeted information needs.

Problem Formulation - Multi-class Classification



[ASE'16, Xu et al.]

Related works

CNN Model
[ASE'16, Xu et al.]

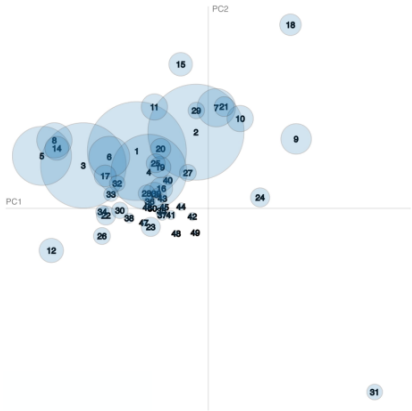
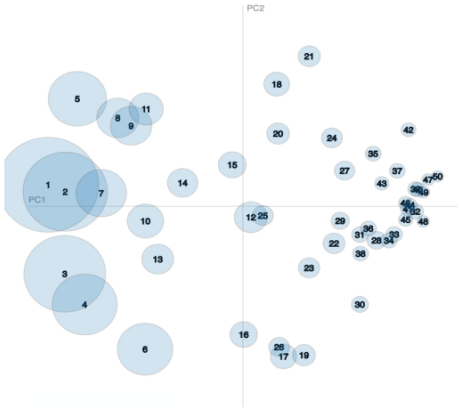
	CNN Model	Basic SVM
Effectiveness	**Win**	Lose
Efficiency (Training time)	Lose	**Win**

Tuning SVM
[FSE'17, Wei Fu and Tim Menzies]

"Try-with-simple" Practice

	CNN Model	**Tuning SVM**
Effectiveness	Lose	**Win**
Efficiency (Training time)	Lose	**Win**

Large Dataset

	Original Dataset [ASE'16, Xu et al.]	Large Dataset
Size (Pairs in total)	6,400	40,000
Quality		

Experiment 1

How well do **CNN Model** and **Tuning SVM** perform on **large dataset**?
Tuning SVM still better?

Effectiveness on Large Dataset

		Duplicate	Direct Link	Indirect Link	Isolated	Overall
Precision	CNN Model	**0.55**	0.33	0.32	**0.79**	**0.50**
	Tuning SVM	0.49	0.33	**0.49**	0.68	0.49
Recall	CNN Model	0.21	**0.62**	0.39	0.41	0.41
	Tuning SVM	**0.59**	0.22	**0.56**	**0.67**	**0.51**
F1-Score	CNN Model	0.31	**0.43**	0.35	0.54	0.41
	Tuning SVM	**0.54**	0.26	**0.52**	**0.68**	**0.50**

Overall, Tuning SVM still outperforms CNN.

Efficiency (Training Cost) on Large Dataset

	Time
CNN Model.	15h 21m 24s
Tuning SVM	38h 24m 46s

CNN Model is more scalable than Tuning SVM.

Q: Why Tuning SVM costs much longer time than CNN Model on the large dataset?

A: With the growth of dataset, Tuning SVM uses a large number of features, adapts several kernels (e.g., RBF kernel) and regularization parameters one by one to tune the SVM.

Experiment 2

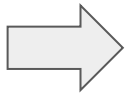
Can we propose another **SVM-based** approach that performs **better** and **faster**?

Soft SVM - A Lightweight Alternative

Soft-Cosine Similarity Measure

Semantic relations (Word Embedding)

Edit-distance based relations



	T_bW_1	T_bW_2	...
T_aW_1	m_{11}	m_{12}	...
T_aW_2	m_{21}	m_{22}	...
...	...	...	...

$= M$

$$Soft\text{-}cosine(a,b) = \frac{\sum\sum_{i,j} N\, a_i m_{ij} b_j}{\sqrt{\sum\sum_{i,j} N\, a_i m_{ij} a_j}\sqrt{\sum\sum_{i,j} N\, b_i m_{ij} b_j}}$$

[SemEval'17, Delphine Charlet and Geraldine Damnati]

Soft SVM vs. Tuning SVM

	Soft SVM	Tuning SVM
Similarity Measurement	Soft-cosine	$Sv=Wv_1 \oplus Wv_2 \oplus \ldots Wv_3$
SVM Kernels	1 (Linear)	4 (Linear, RBF, Poly, Sigmod)

[SemEval'17, Delphine Charlet and Geraldine Damnati]

Effectiveness of Soft SVM

		Duplicate	Direct Link	Indirect Link	Isolated	Overall
Precision	CNN Model	**0.55**	0.33	0.32	**0.79**	0.50
	Tuning SVM	0.49	0.33	**0.49**	0.68	0.49
	Soft SVM	0.51	**0.45**	0.42	0.75	**0.53**
Recall	CNN Model	0.21	**0.62**	0.39	0.41	0.41
	Tuning SVM	**0.59**	0.22	0.56	0.67	0.51
	Soft SVM	0.48	0.21	**0.58**	**0.90**	**0.54**
F1-Score	CNN Model	0.31	**0.43**	0.35	0.54	0.41
	Tuning SVM	**0.54**	0.26	**0.52**	0.68	0.50
	Soft SVM	0.50	0.29	0.49	**0.82**	**0.52**

Overall, Soft SVM achieves best performance.

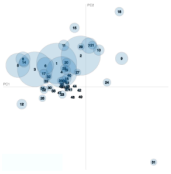
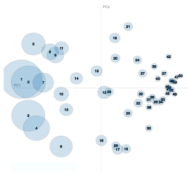
Efficiency (Training Cost) of Soft SVM

	Time
CNN Model.	15h 21m 24s
Tuning SVM	38h 24m 46s
Soft SVM	**2h 54m**

Soft SVM costs much less time than CNN Model (5x) and Tuning SVM (13x)!

Conclusion

Large Dataset

	Original Dataset [ASE'16, Xu et al.]	Large Dataset
Size (Pairs in total)	6,400	40,000
Quality		

Efficiency (Training Cost) on Large Dataset

	Time
CNN Model.	15h 21m 24s
Tuning SVM	38h 24m 46s

CNN Model is more scalable than Tuning SVM.

Effectiveness of Soft SVM

		Duplicate	Direct Link	Indirect Link	Isolated	Overall
Precision	CNN Model	**0.55**	0.33	0.32	**0.79**	0.50
	Tuning SVM	0.49	0.33	**0.49**	0.68	0.49
	Soft SVM	0.51	**0.45**	0.42	0.75	**0.53**
Recall	CNN Model	0.21	**0.62**	0.39	0.41	0.41
	Tuning SVM	**0.59**	0.22	0.56	0.67	0.51
	Soft SVM	0.48	0.21	**0.58**	**0.90**	**0.54**
F1-Score	CNN Model	0.31	**0.43**	0.35	0.54	0.41
	Tuning SVM	**0.54**	0.26	**0.52**	0.68	0.50
	Soft SVM	0.50	0.29	0.49	**0.82**	**0.52**

Overall, Soft SVM achieves best performance.

Efficiency (Training Cost) of Soft SVM

	Time
CNN Model.	15h 21m 24s
Tuning SVM	38h 24m 46s
Soft SVM	**2h 54m**

Soft SVM costs much less time than CNN Model (5x) and Tuning SVM (13x)!

Implication

1. “Try-with-simple” practice still holds.
2. Larger dataset may uncover new limitations.
3. The performance of the task *Prediction of Relatedness in Stack Overflow* still has room to improve. Consideration of more features (e.g., tags) may be an easier way to do it.

Thanks & QA!

All experiment data can be found here:
https://github.com/XBWer/ESEM2018