NC STATE UNIVERSITY

... if ~~λ~~ software engineering then NC State ...

Easy over Hard: A Case Study on Deep Learning

Wei Fu
wfu@ncsu.edu
http://weifu.us

Tim Menzies
tim.menzies@gmail.com
http://menzies.us

Sep, 2017

On the market

(expected graduation: May,2018)

weifu.us

Tim Menzies
(advisor)

Research Areas:

- Machine learning
- SBSE
- Evolutionary algorithms
- Hyper-parameter tuning

Publications:

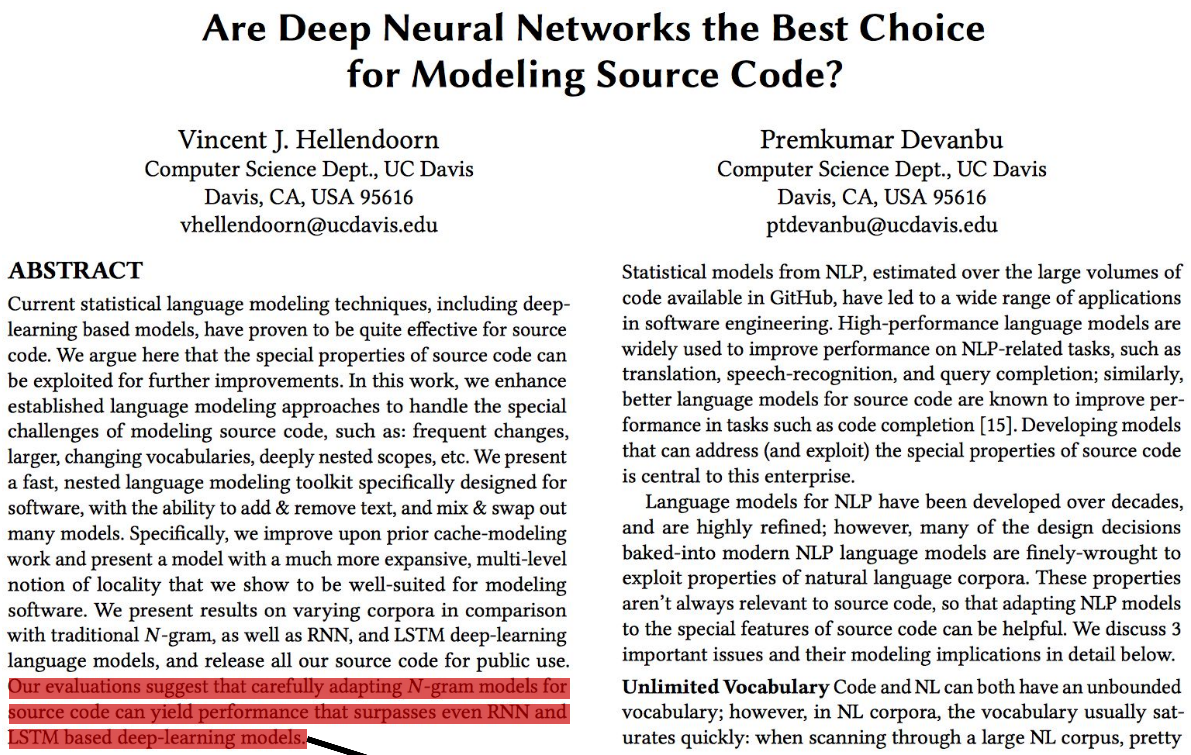
FSE: 2
ASE: 1
TSE: 1
IST: 1
Under Review: 2

Q: Do we need deep learning for software analytics?

Q: Do we need deep learning for software analytics?

A: Maybe not

Another FSE'17 Paper on Deep Learning



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.

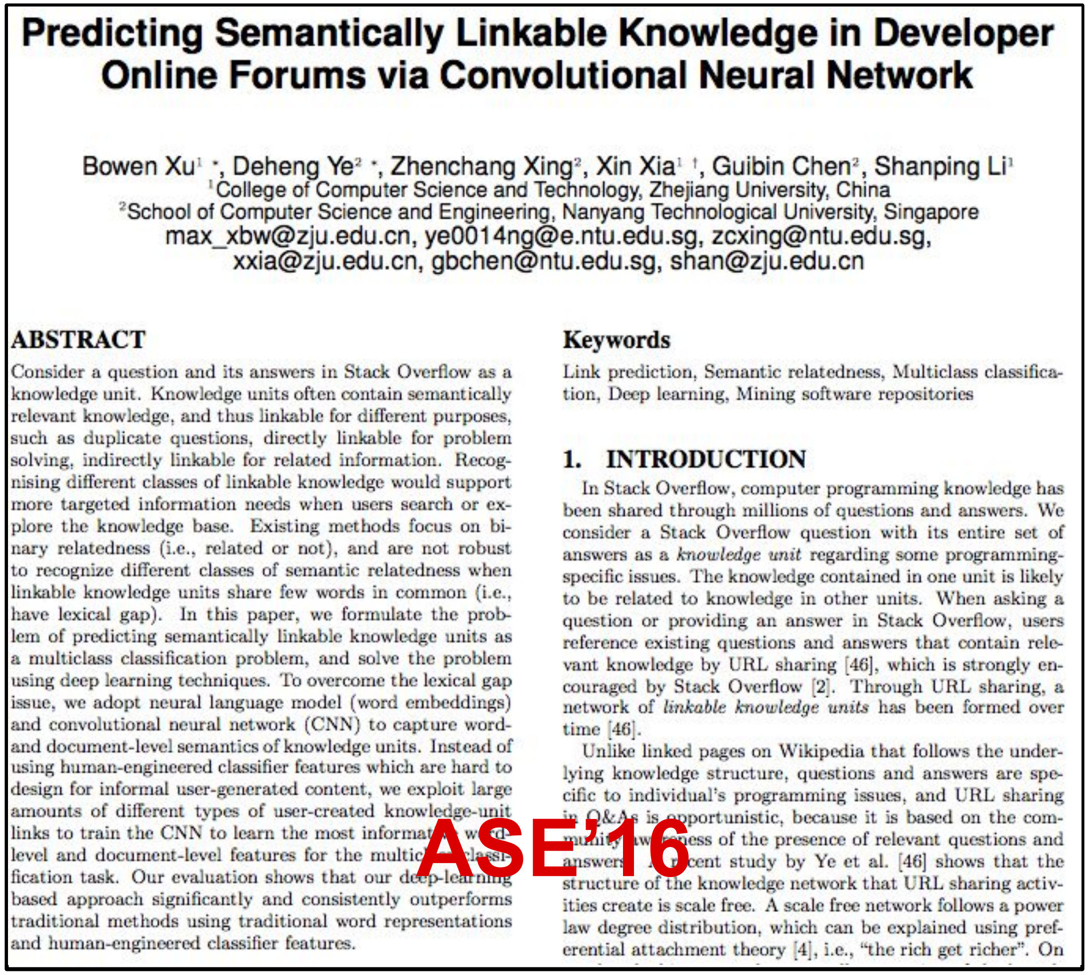
Statistical models from NLP, estimated over the large volumes of code available in GitHub, have led to a wide range of applications in software engineering. High-performance language models are widely used to improve performance on NLP-related tasks, such as translation, speech-recognition, and query 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

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.

We Promote Open Science



Predicting Semantically Linkable Knowledge in Developer Online Forums via Convolutional Neural Network

Bowen Xu[1] *, Deheng Ye[2] *, Zhenchang Xing[2], Xin Xia[1] †, Guibin Chen[2], Shanping Li[1]
[1]College of Computer Science and Technology, Zhejiang University, China
[2]School of Computer Science and Engineering, Nanyang Technological University, Singapore
max_xbw@zju.edu.cn, ye0014ng@e.ntu.edu.sg, zcxing@ntu.edu.sg, xxia@zju.edu.cn, gbchen@ntu.edu.sg, shan@zju.edu.cn

ABSTRACT

Consider a question and its answers in Stack Overflow as a knowledge unit. Knowledge units often contain semantically relevant knowledge, and thus linkable for different purposes, such as duplicate questions, directly linkable for problem solving, indirectly linkable for related information. Recognising different classes of linkable knowledge would support more targeted information needs when users search or explore the knowledge base. Existing methods focus on binary relatedness (i.e., related or not), and are not robust to recognize different classes of semantic relatedness when linkable knowledge units share few words in common (i.e., have lexical gap). In this paper, we formulate the problem of predicting semantically linkable knowledge units as a multiclass classification problem, and solve the problem using deep learning techniques. To overcome the lexical gap issue, we adopt neural language model (word embeddings) and convolutional neural network (CNN) to capture word- and document-level semantics of knowledge units. Instead of using human-engineered classifier features which are hard to design for informal user-generated content, we exploit large amounts of different types of user-created knowledge-unit links to train the CNN to learn the most informative word-level and document-level features for the multiclass classification task. Our evaluation shows that our deep-learning based approach significantly and consistently outperforms traditional methods using traditional word representations and human-engineered classifier features.

Keywords

Link prediction, Semantic relatedness, Multiclass classification, Deep learning, Mining software repositories

1. INTRODUCTION

In Stack Overflow, computer programming knowledge has been shared through millions of questions and answers. We consider a Stack Overflow question with its entire set of answers as a *knowledge unit* regarding some programming-specific issues. The knowledge contained in one unit is likely to be related to knowledge in other units. When asking a question or providing an answer in Stack Overflow, users reference existing questions and answers that contain relevant knowledge by URL sharing [46], which is strongly encouraged by Stack Overflow [2]. Through URL sharing, a network of *linkable knowledge units* has been formed over time [46].

Unlike linked pages on Wikipedia that follows the underlying knowledge structure, questions and answers are specific to individual's programming issues, and URL sharing in Q&As is opportunistic, because it is based on the community awareness of the presence of relevant questions and answers. A recent study by Ye et al. [46] shows that the structure of the knowledge network that URL sharing activities create is scale free. A scale free network follows a power law degree distribution, which can be explained using preferential attachment theory [4], i.e., "the rich get richer". On

ASE'16

THANKS!

Make data available

Our code & data at:

https://github.com/WeiFoo/easyOverHard

Faster or More Complex SE Analytics?

More complex methods:

- New feature selection[1].
- New feature discovering[2].

Costs:

- Learn control settings: weeks to years[3,4]
- Deep learning:
  - Lam et al.: weeks of CPU[6].
  - Gu et al.:240 hours of GPU[13].

Fisher et al[5]:

- Large quantities of low value data to small set of higher value data.
- Luxuries of interactivity, direct manipulation and fast system response are gone.

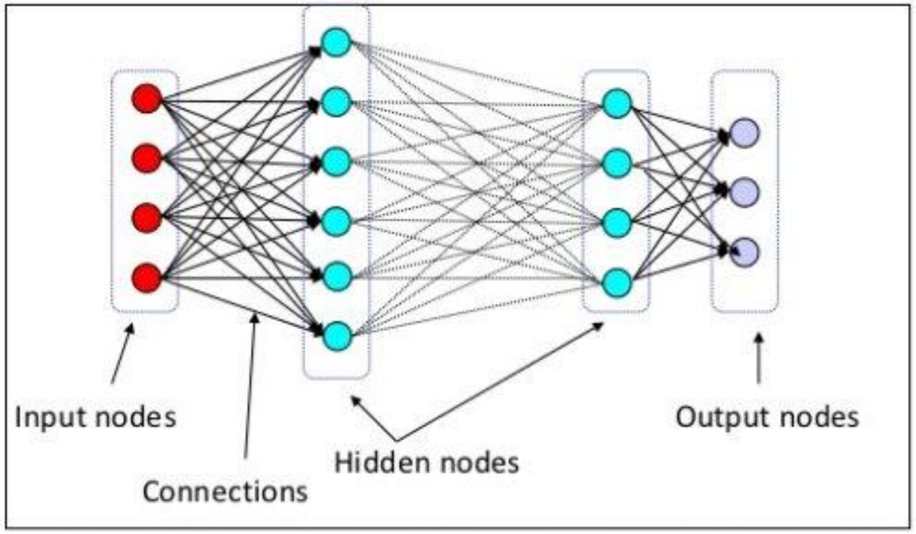
Faster vs. More Complex SE Analytics

Does the improvement worth the cost ?

A case study on deep learning

Deep Learning

- Built on multiple layers of neural networks.
- Composing simple but non-linear modules to explore high-dimensional data [42].

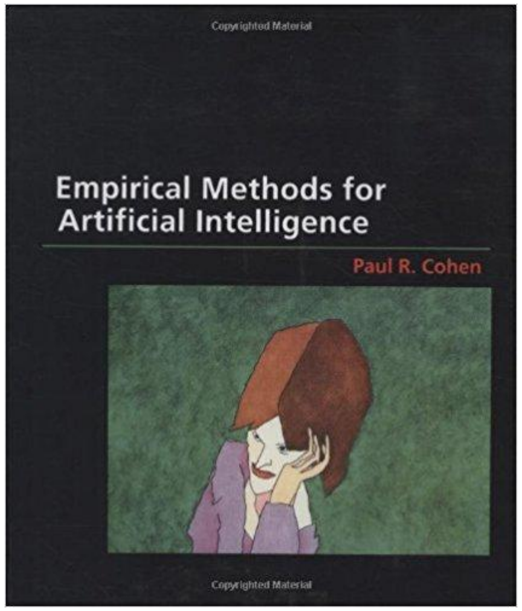


Deep Learning in SE

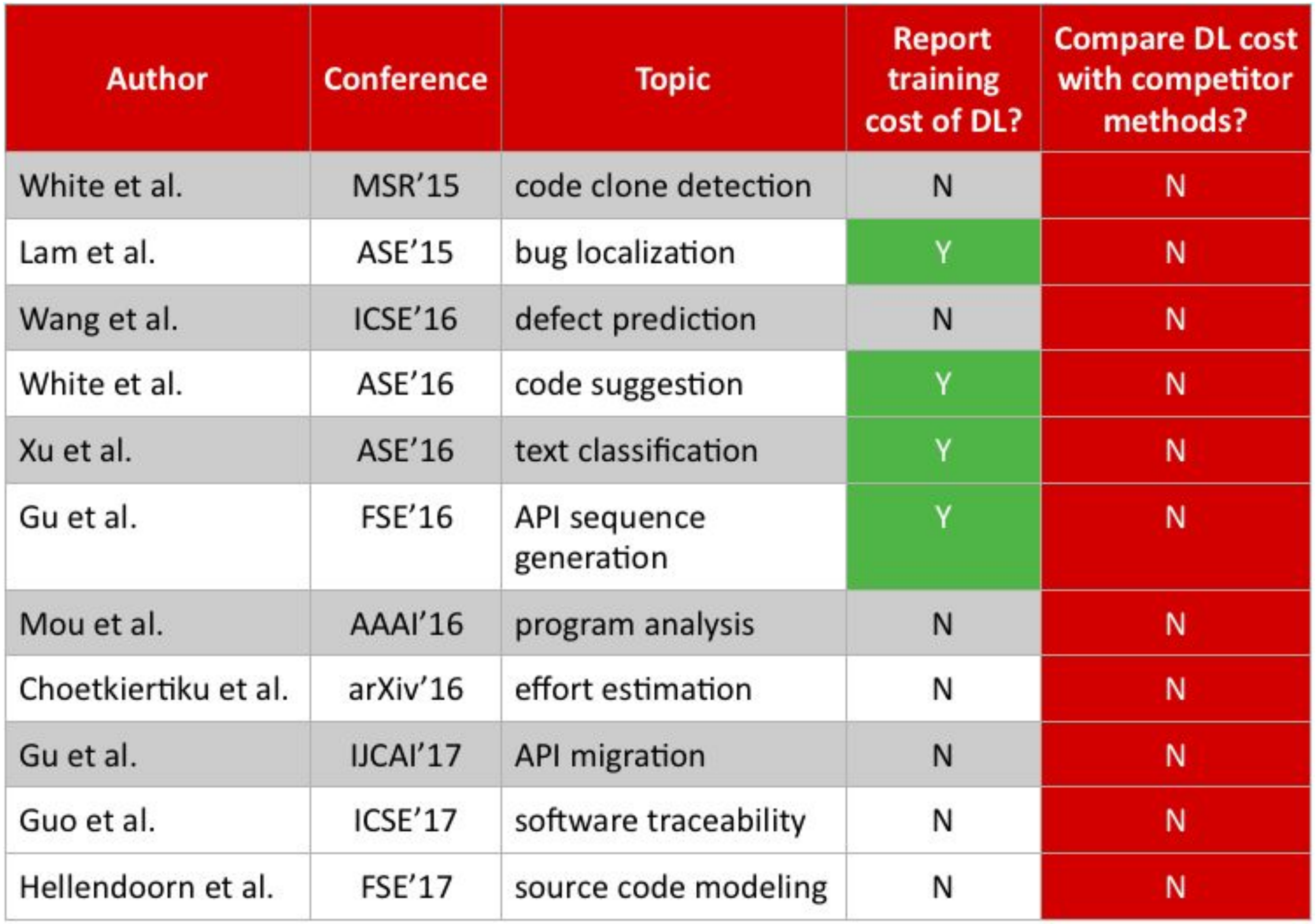
Author	Conference	Topic
White et al.	MSR'15	code clone detection
Lam et al.	ASE'15	bug localization
Wang et al.	ICSE'16	defect prediction
White et al.	ASE'16	code suggestion
Xu et al.	ASE'16	text classification
Gu et al.	FSE'16	API sequence generation
Mou et al.	AAAI'16	program analysis
Choetkiertiku et al.	arXiv'16	effort estimation
Gu et al.	IJCAI'17	API migration
Guo et al.	ICSE'17	software traceability
Hellendoorn et al.	FSE'17	source code modeling

Deep Learning in SE

Author	Conference	Topic	Report training cost of DL?
White et al.	MSR'15	code clone detection	N
Lam et al.	ASE'15	bug localization	Y
Wang et al.	ICSE'16	defect prediction	N
White et al.	ASE'16	code suggestion	Y
Xu et al.	ASE'16	text classification	Y
Gu et al.	FSE'16	API sequence generation	Y
Mou et al.	AAAI'16	program analysis	N
Choetkiertiku et al.	arXiv'16	effort estimation	N
Gu et al.	IJCAI'17	API migration	N
Guo et al.	ICSE'17	software traceability	N
Hellendoorn et al.	FSE'17	source code modeling	N



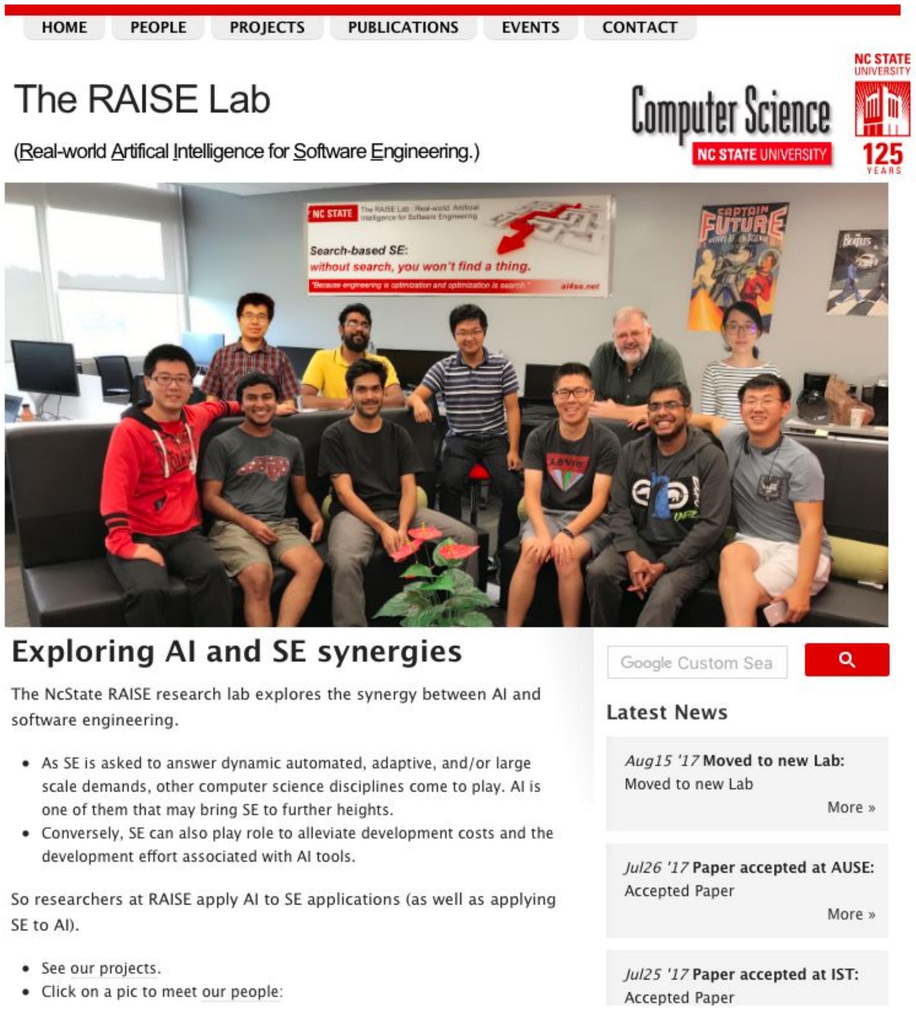
Deep Learning in SE



Author	Conference	Topic	Report training cost of DL?	Compare DL cost with competitor methods?
White et al.	MSR'15	code clone detection	N	N
Lam et al.	ASE'15	bug localization	Y	N
Wang et al.	ICSE'16	defect prediction	N	N
White et al.	ASE'16	code suggestion	Y	N
Xu et al.	ASE'16	text classification	Y	N
Gu et al.	FSE'16	API sequence generation	Y	N
Mou et al.	AAAI'16	program analysis	N	N
Choetkiertiku et al.	arXiv'16	effort estimation	N	N
Gu et al.	IJCAI'17	API migration	N	N
Guo et al.	ICSE'17	software traceability	N	N
Hellendoorn et al.	FSE'17	source code modeling	N	N

Trade-off: Benefit vs. Cost ?

Faster Software Analytics



HOME PEOPLE PROJECTS PUBLICATIONS EVENTS CONTACT

The RAISE Lab

(Real-world Artifical Intelligence for Software Engineering.)

Exploring AI and SE synergies

The NcState RAISE research lab explores the synergy between AI and software engineering.

- As SE is asked to answer dynamic automated, adaptive, and/or large scale demands, other computer science disciplines come to play. AI is one of them that may bring SE to further heights.
- Conversely, SE can also play role to alleviate development costs and the development effort associated with AI tools.

So researchers at RAISE apply AI to SE applications (as well as applying SE to AI).

- See our projects.
- Click on a pic to meet our people:

Google Custom Sea

Latest News

Aug15 '17 **Moved to new Lab:** Moved to new Lab
More »

Jul26 '17 **Paper accepted at AUSE:** Accepted Paper
More »

Jul25 '17 **Paper accepted at IST:** Accepted Paper

Reproducible, Faster method!

Method

Case Study

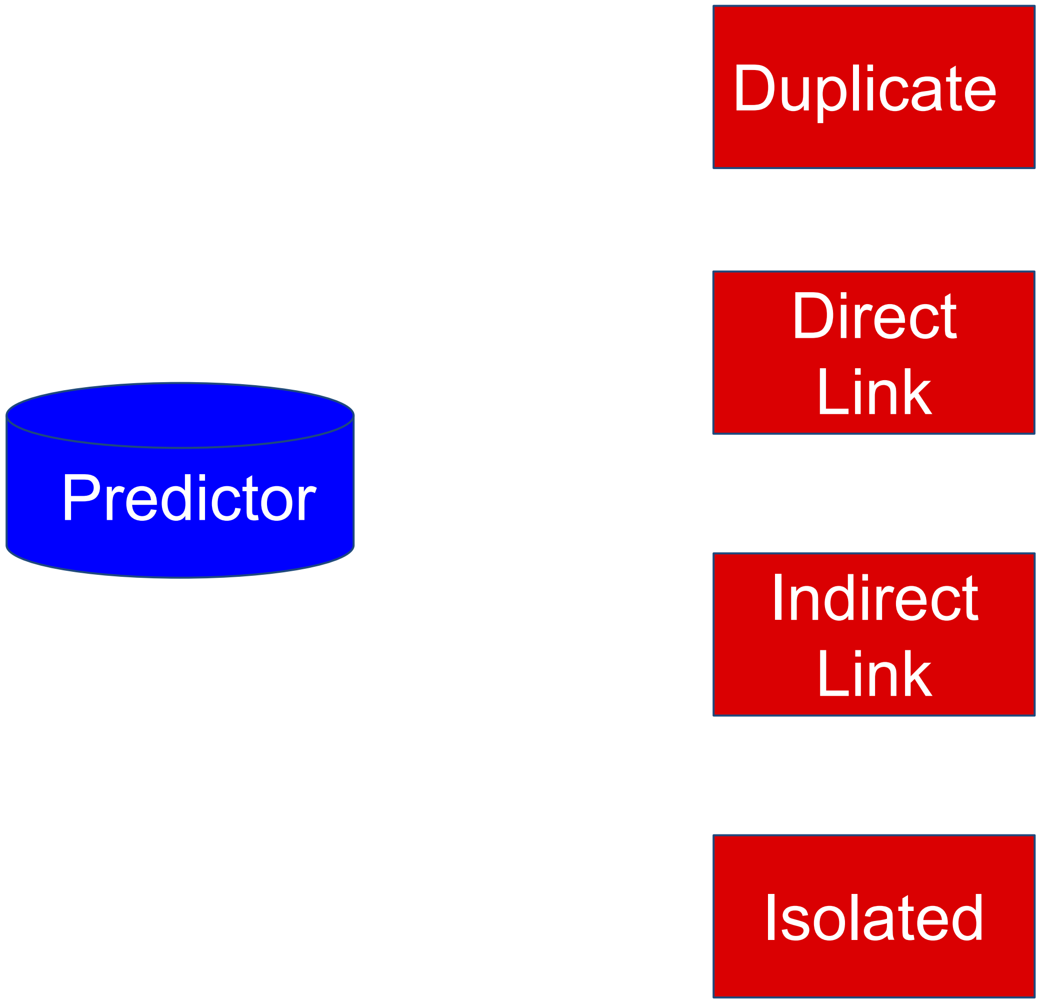
Linkable Questions Prediction on StackOverflow

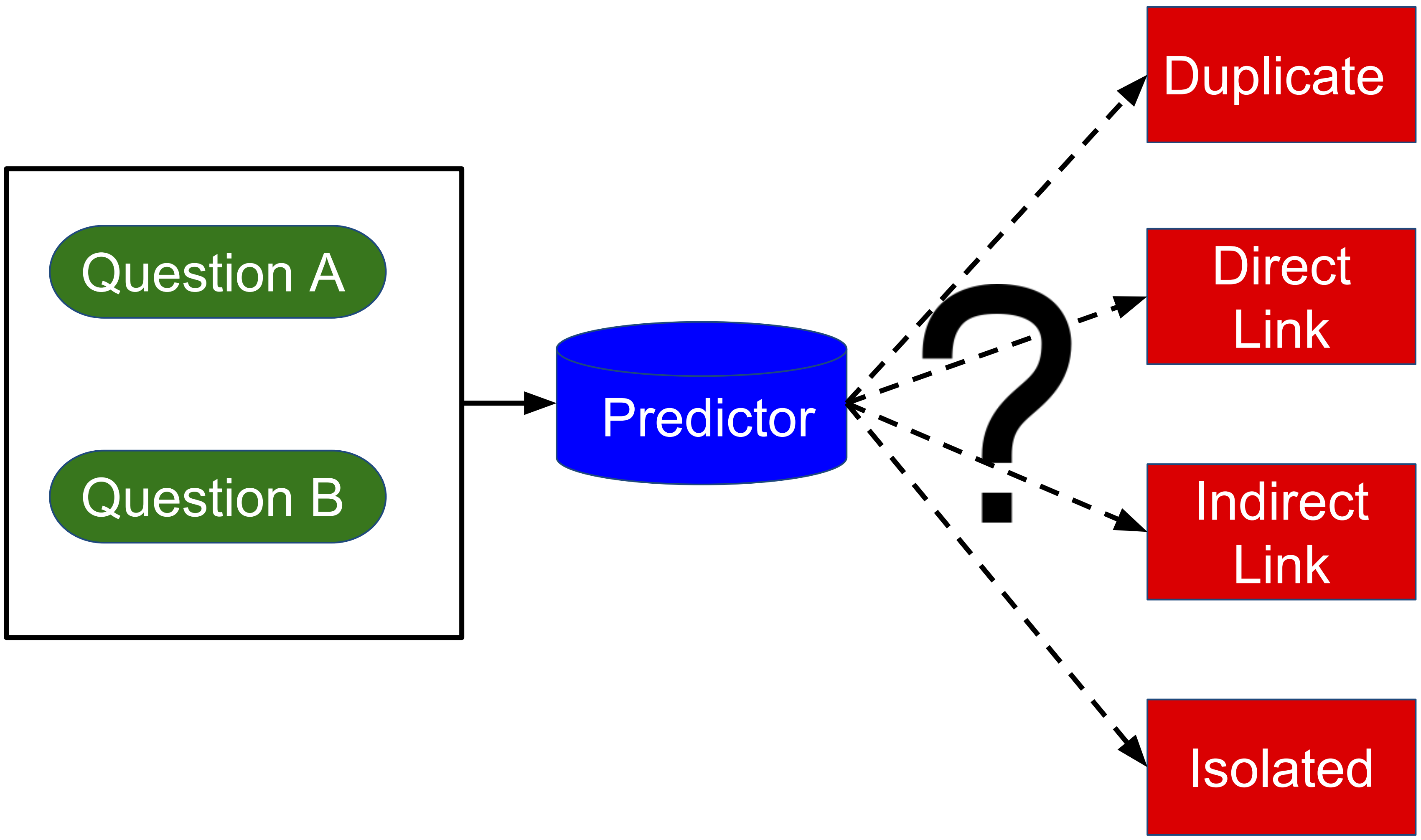
Duplicate

Direct Link

Indirect Link

Isolated


Duplicate
Direct Link
Predictor
Indirect Link
Isolated


Question A
Question B
Predictor
?
Duplicate
Direct Link
Indirect Link
Isolated

Learners

- Baseline:
  - SVM
- Xu's deep learning method:
  - CNN(convolutional neural networks)
- Our proposed method:
  - SVM + DE

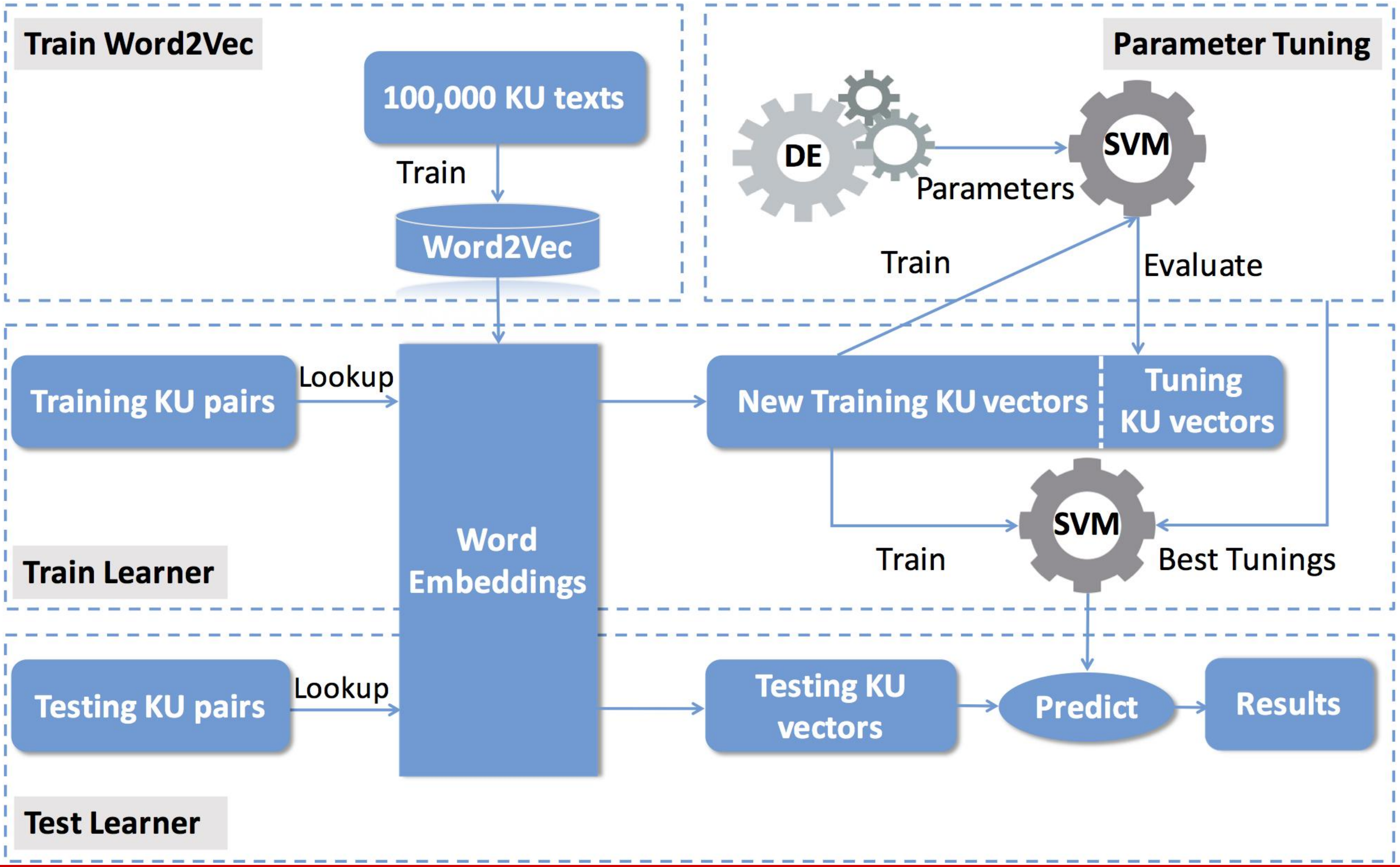
Differential Evolution for Hyper-parameter Tuning

Tuning Algorithm: Differential Evolution*

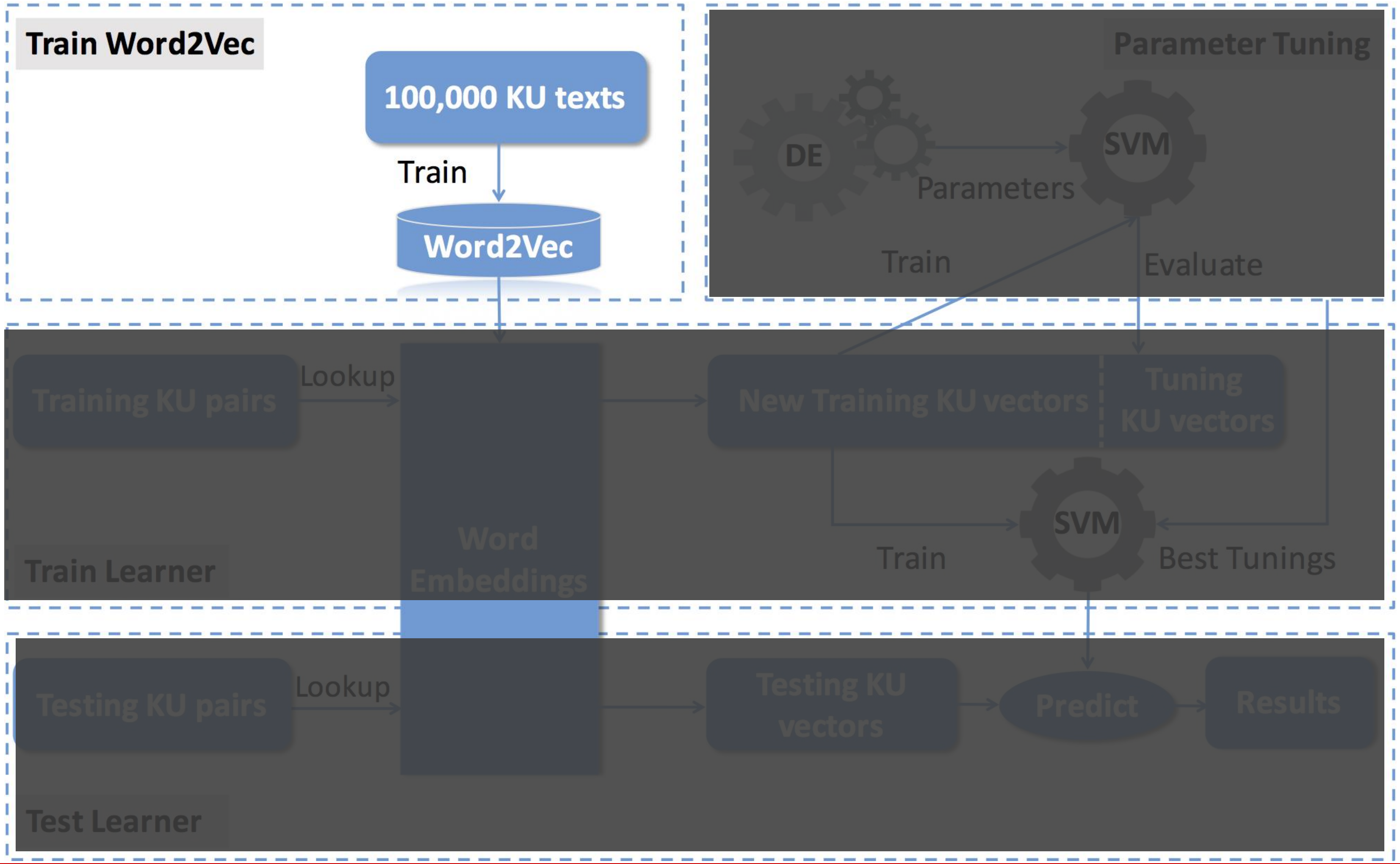
```
Frontier = Pick N options at random # e.g. N =10
M times repeat : # e.g. M = 5
    for Parent in Frontier:
        ● Select a, b, c = three other frontier items.
        ● Candidate = a + f*(b-c) # ish
        ● if Candidate "better", replace Parent.
```

* Storn, Rainer, and Kenneth Price. "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces." *Journal of global optimization* 11.4 (1997): 341-359.

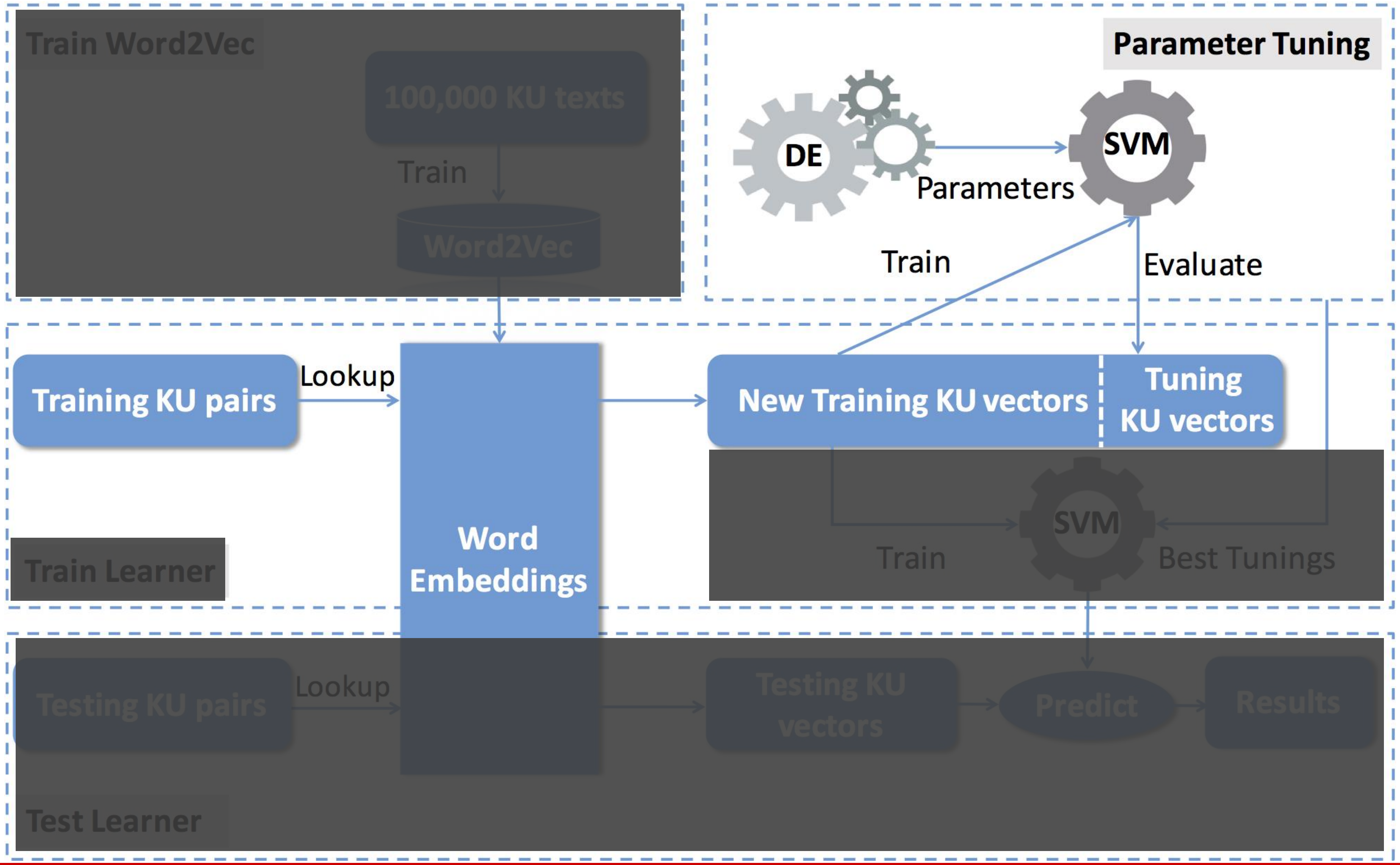
Experimental Setup



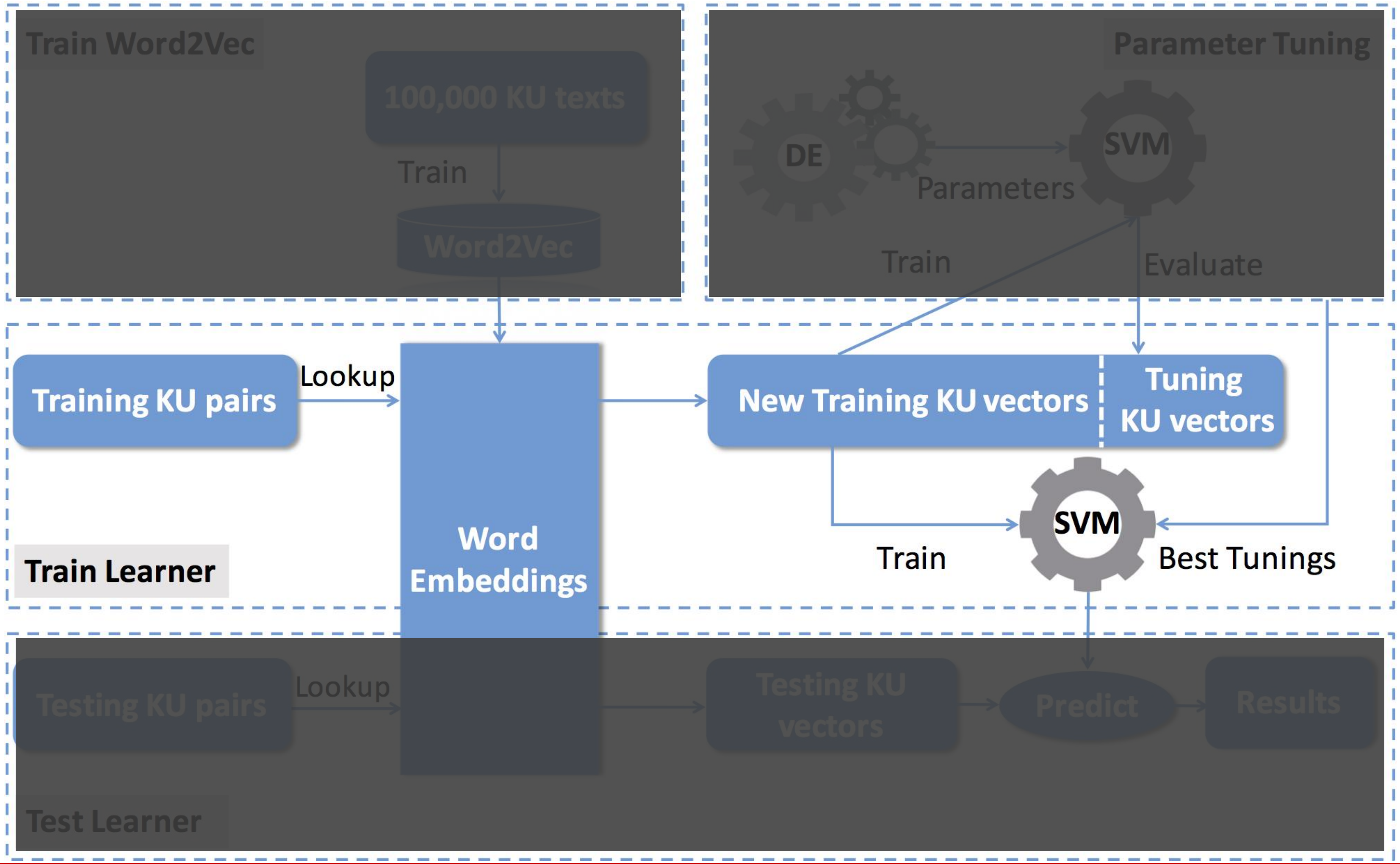
Experimental Setup



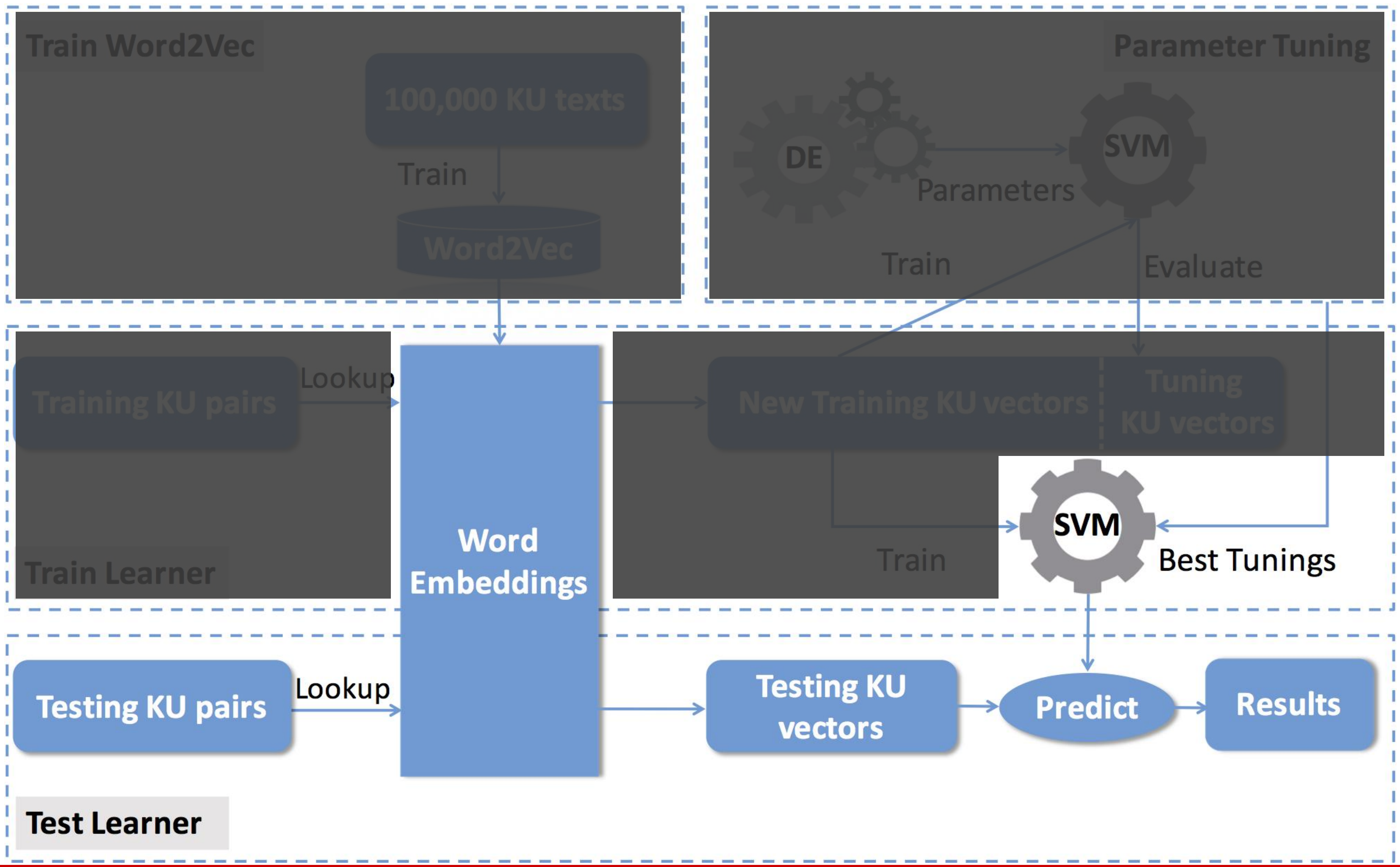
Experimental Setup



Experimental Setup



Experimental Setup



Results

Research Questions

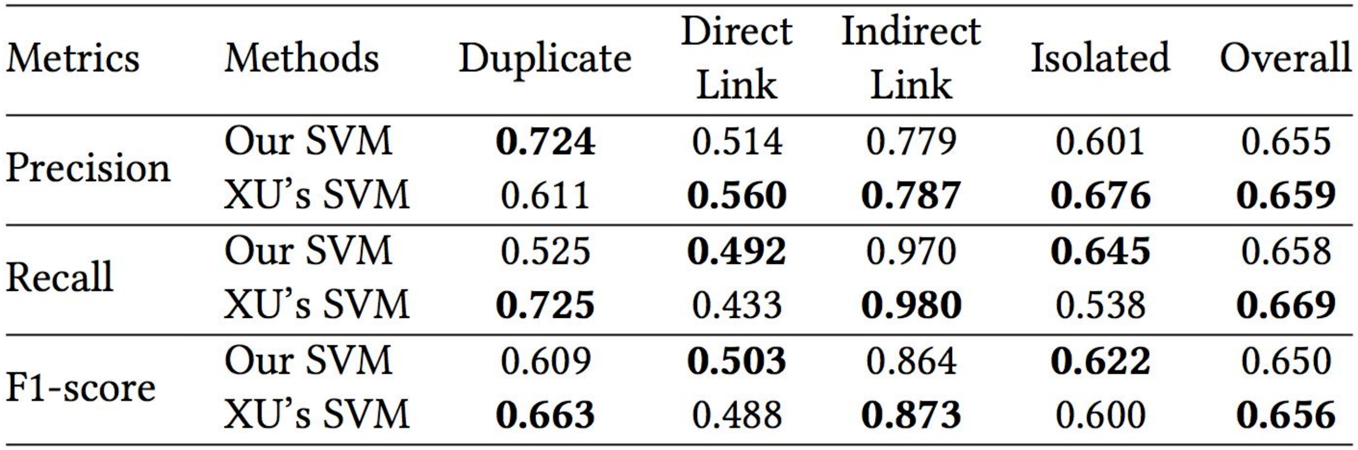
RQ1: Can we reproduce Xu’s baseline results?

RQ2: DE+SVM outperforms Xu’s deep learning method?

RQ3: DE+SVM faster than Xu’s deep learning method?

RQ1: Reproduce Xu’s Baseline Results?

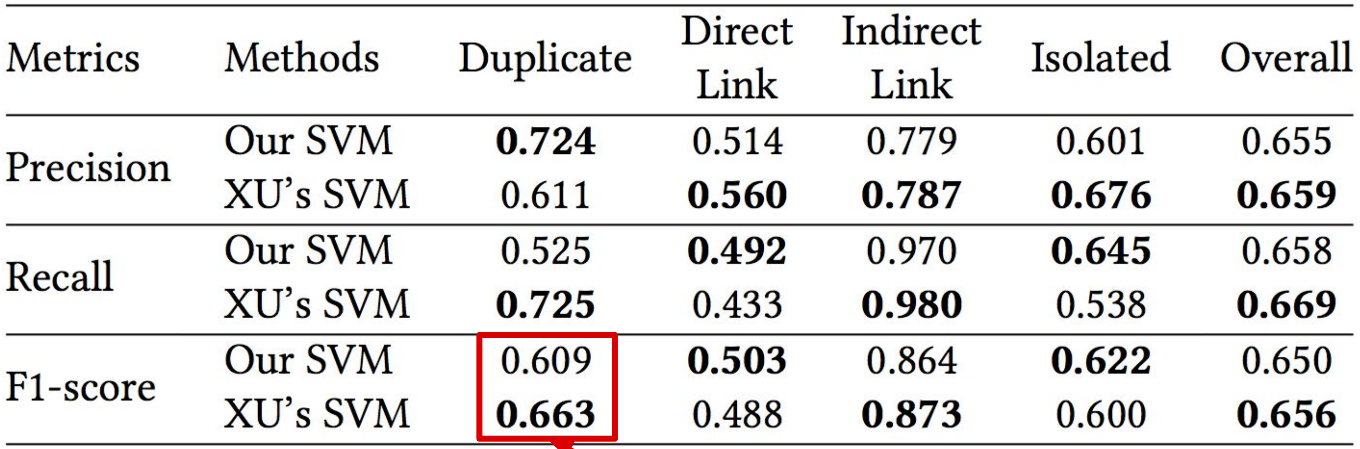
Comparison of our baseline method with Xu’s baseline. Best scores are marked in bold.



Metrics	Methods	Duplicate	Direct Link	Indirect Link	Isolated	Overall
Precision	Our SVM	**0.724**	0.514	0.779	0.601	0.655
	XU’s SVM	0.611	**0.560**	**0.787**	**0.676**	**0.659**
Recall	Our SVM	0.525	**0.492**	0.970	**0.645**	0.658
	XU’s SVM	**0.725**	0.433	**0.980**	0.538	**0.669**
F1-score	Our SVM	0.609	**0.503**	0.864	**0.622**	0.650
	XU’s SVM	**0.663**	0.488	**0.873**	0.600	**0.656**

RQ1: Reproduce Xu’s Baseline Results?

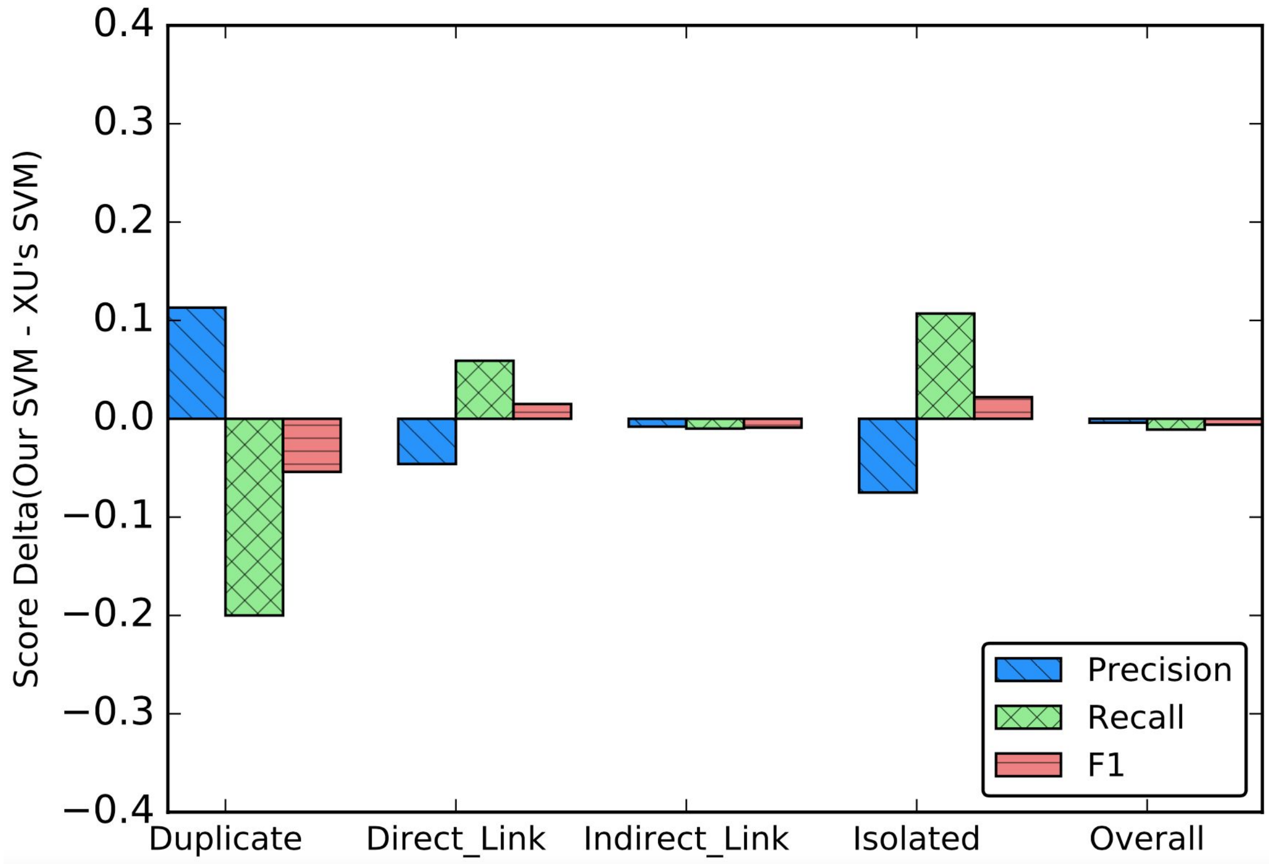
Comparison of our baseline method with Xu’s baseline. Best scores are marked in bold.



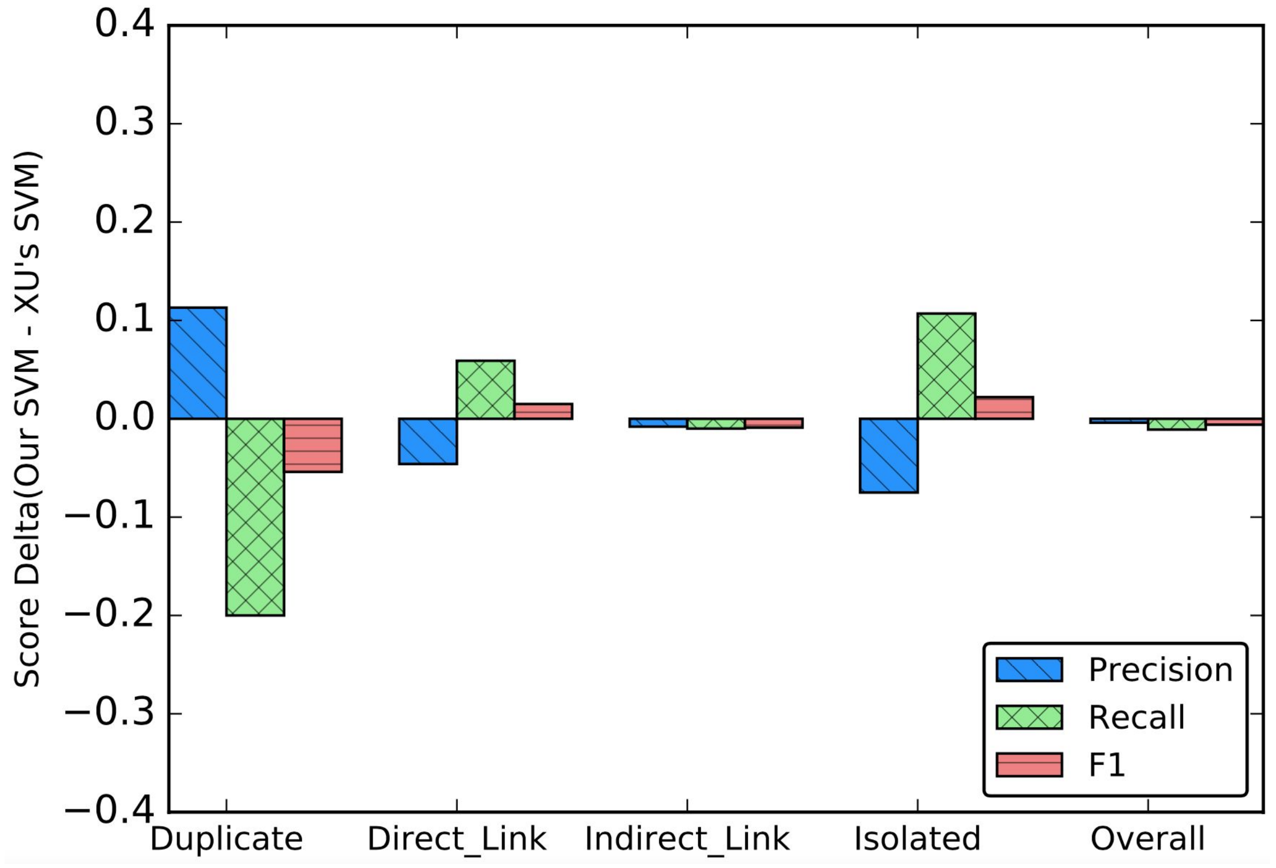
Metrics	Methods	Duplicate	Direct Link	Indirect Link	Isolated	Overall
Precision	Our SVM	**0.724**	0.514	0.779	0.601	0.655
	XU’s SVM	0.611	**0.560**	**0.787**	**0.676**	**0.659**
Recall	Our SVM	0.525	**0.492**	0.970	**0.645**	0.658
	XU’s SVM	**0.725**	0.433	**0.980**	0.538	**0.669**
F1-score	Our SVM	0.609	**0.503**	0.864	**0.622**	0.650
	XU’s SVM	**0.663**	0.488	**0.873**	0.600	**0.656**

Score Delta(F1) = Our SVM - Xu’s SVM = -0.054

RQ1: Reproduce Xu's Baseline Results?



RQ1: Reproduce Xu's Baseline Results?



Overall, we got similar results to the baseline method reported in XU's study

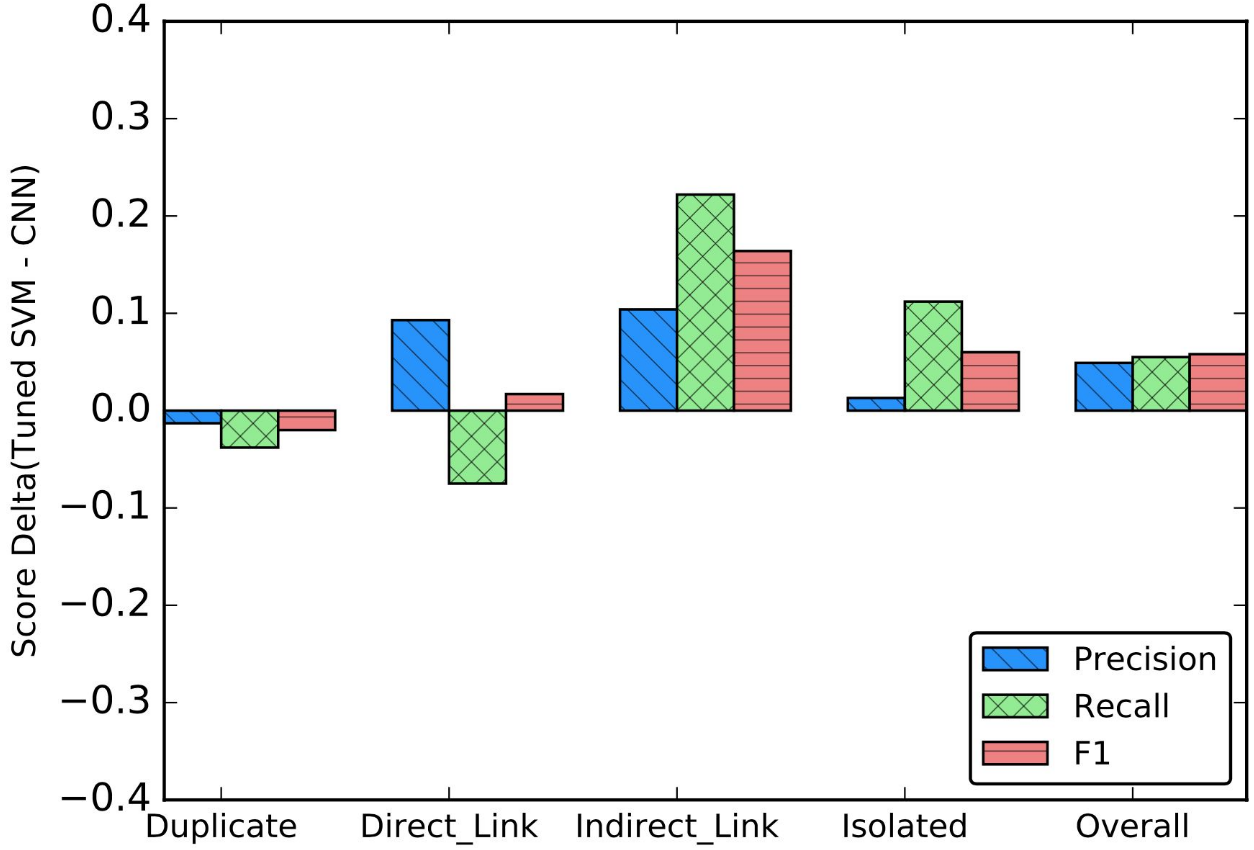
Research Questions

RQ1: Can we reproduce Xu’s baseline results?

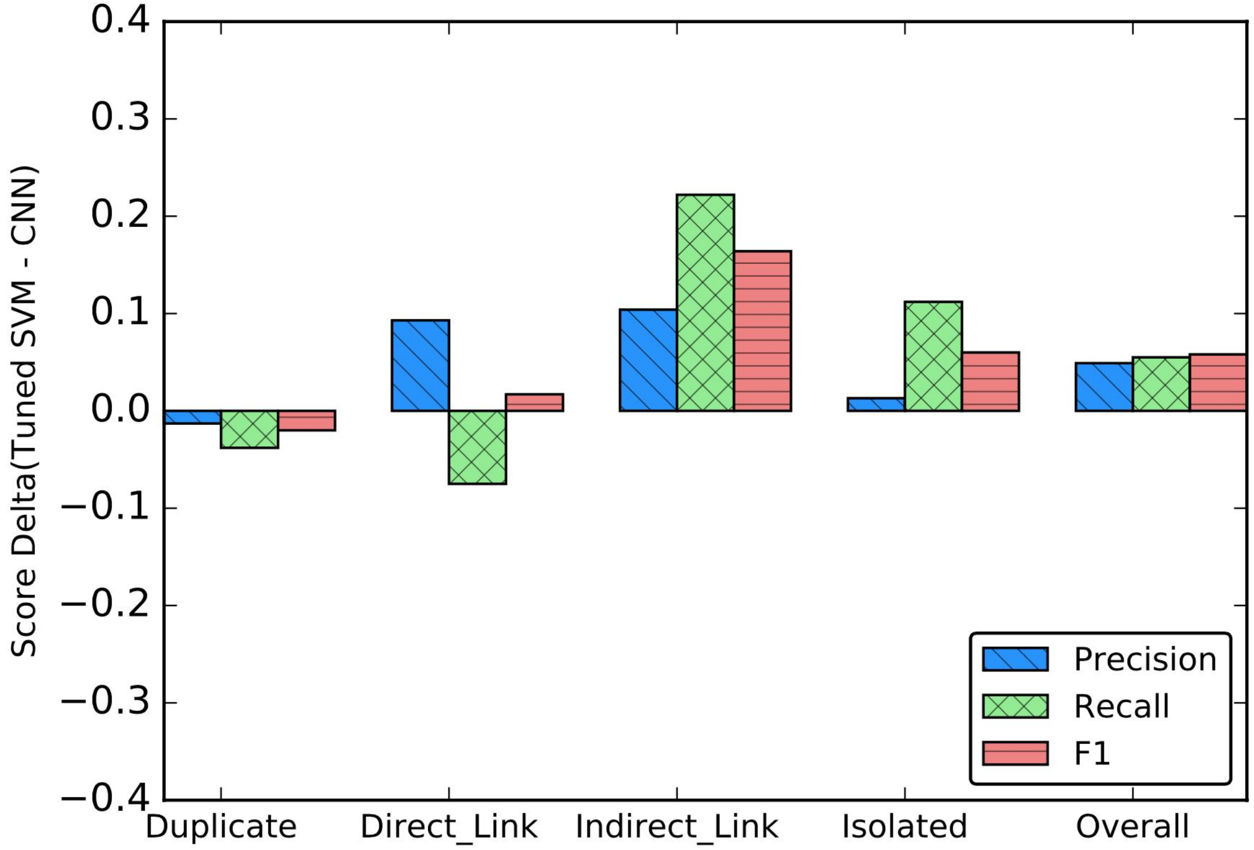
RQ2: DE+SVM outperforms Xu’s deep learning method?

RQ3: DE+SVM faster than Xu’s deep learning method?

RQ2: DE+SVM Outperforms Xu's CNN?



RQ2: DE+SVM Outperforms Xu's CNN?



Deep learning(CNN) does not have any performance advantage over DE+SVM.

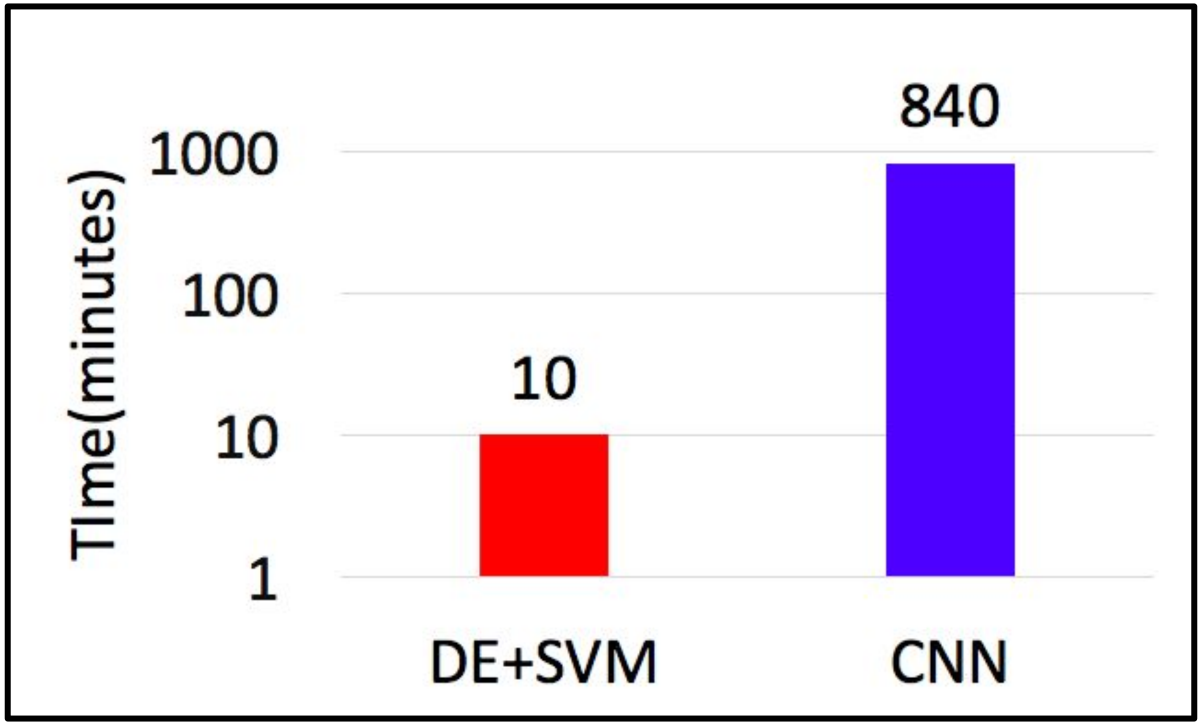
Research Questions

RQ1: Can we reproduce Xu’s baseline results?

RQ2: DE+SVM outperforms Xu’s deep learning method?

RQ3: DE+SVM faster than Xu’s deep learning method?

RQ3: Faster than Xu’s CNN?



DE+SVM is 84X faster than deep learning(CNN) in terms of model building.

Conclusion

Observation

Simple DE tuning performs

Better & Faster than deep learning!

Q: Do we still need deep learning for software analytics?

Q: Do we still need deep learning for software analytics?

A: Maybe not

Q: Do we *still* need deep learning for software analytics?

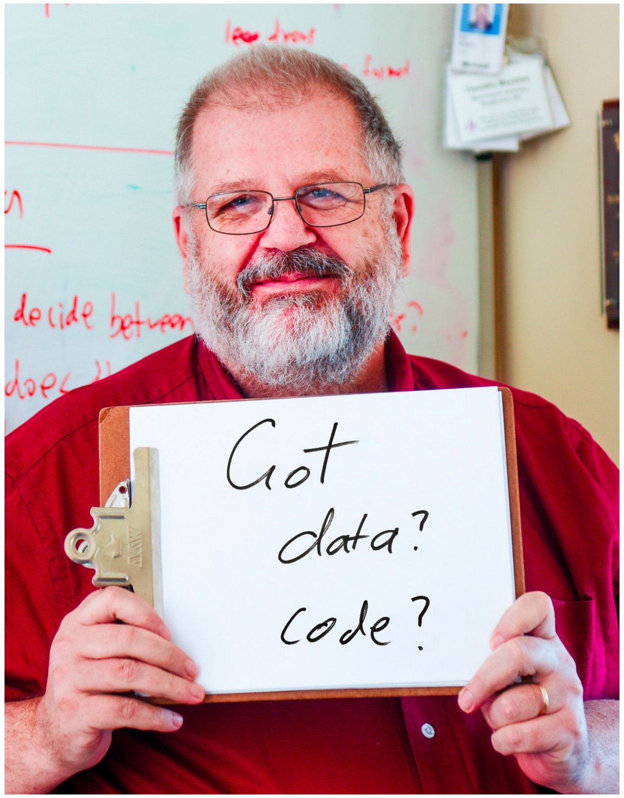
A: Maybe not

Easy first before hard

Implication

For future deep learning in SE:

- **FINE TUNE** your baseline methods
- Do not ignore the **COST** of deep learning.
- **SHARE** your code and data if possible



tiny.cc/seacraft

Why Seacraft?

- Successor of PROMISE repo, which contains a lot of SE artifacts.
- No data limits;
- Provides DOI for every submission (aka, easy citation);
- Automatic updates if linked to github project.

On the market

*(expected graduation:**May,2018**)*

weifu.us

(Machine learning, SBSE, Evolutionary algorithms, Hyper-parameter tuning)

Easy over hard

Publications:

FSE: 2
ASE: 1
TSE: 1
IST: 1
Under Review: 2

Reference

1. Hall, Mark A., and Geoffrey Holmes. "Benchmarking attribute selection techniques for discrete class data mining." *IEEE Transactions on Knowledge and Data engineering* 15.6 (2003): 1437-1447.
2. Jiang, Tian, Lin Tan, and Sunghun Kim. "Personalized defect prediction." *Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering*. IEEE Press, 2013.
3. Fu, Wei, Vivek Nair, and Tim Menzies. "Why is Differential Evolution Better than Grid Search for Tuning Defect Predictors?." *arXiv preprint arXiv:1609.02613* (2016).
4. Wang, Tiantian, et al. "Searching for better configurations: a rigorous approach to clone evaluation." *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*. ACM, 2013.
5. Fisher, Danyel, et al. "Interactions with big data analytics." *interactions* 19.3 (2012): 50-59.
6. [Lam ASE'15]Lam, An Ngoc, et al. "Combining deep learning with information retrieval to localize buggy files for bug reports (n)." *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*. IEEE, 2015.
7. [Wang ICSE'16]Wang, Song, Taiyue Liu, and Lin Tan. "Automatically learning semantic features for defect prediction." *Proceedings of the 38th International Conference on Software Engineering*. ACM, 2016.
8. [White MSR'15]White, Martin, et al. "Toward deep learning software repositories." *Mining Software Repositories (MSR), 2015 IEEE/ACM 12th Working Conference on*. IEEE, 2015.
9. [White ASE'15]White, Martin, et al. "Deep learning code fragments for code clone detection." *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*. ACM, 2016.
10. [Xu ASE'16]Xu, Bowen, et al. "Predicting semantically linkable knowledge in developer online forums via convolutional neural network." *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*. ACM, 2016.
11. [Yuan 2014]Yuan, Zhenlong, et al. "Droid-Sec: deep learning in android malware detection." *ACM SIGCOMM Computer Communication Review*. Vol. 44. No. 4. ACM, 2014.
12. [Mou AAAI'2016]Mou, Lili, et al. "Convolutional Neural Networks over Tree Structures for Programming Language Processing." *AAAI*. 2016.
13. [Gu FSE'16]Gu, Xiaodong, et al. "Deep API learning." *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 2016.
14. [Gu IJCAI'17]Gu, Xiaodong, et al. "DeepAM: Migrate APIs with Multi-modal Sequence to Sequence Learning." *arXiv preprint arXiv:1704.07734* (2017).
15. [Choetkiertikul arXiv'16]Choetkiertikul, Morakot, et al. "A deep learning model for estimating story points." *arXiv preprint arXiv:1609.00489* (2016).

Reference

16. [Fu arXiv'16]Fu, Wei, Vivek Nair, and Tim Menzies. "Why is Differential Evolution Better than Grid Search for Tuning Defect Predictors?." *arXiv preprint arXiv:1609.02613* (2016).
17. [Fu IST'16]Fu, Wei, Tim Menzies, and Xipeng Shen. "Tuning for software analytics: Is it really necessary?." *Information and Software Technology* 76 (2016): 135-146.
18. [Hellendoorn FSE'17]Hellendoorn, Vincent J., and Premkumar Devanbu. "Are deep neural networks the best choice for modeling source code?." *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. ACM, 2017.