ai4se.net

Faster methods for software analytics!

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

Find these slides at: http://tiny.cc/wfuExam17

Sep, 2017

software analytics

software analytics

To enable software practitioners to perform data exploration and analysis in order to obtain insightful and actionable information for data-driven tasks around software.

Effort estimation

Performance modeling

Defect prediction

Linkable questions prediction

Issue closing time prediction

software analytics

To enable software practitioners to perform data exploration and analysis in order to obtain insightful and actionable information for data-driven tasks around software.

Defect prediction

Linkable questions prediction

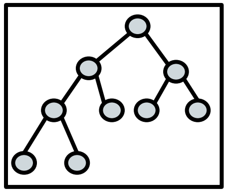
software analytics

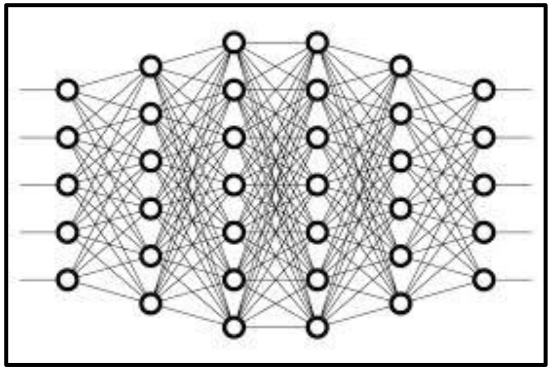
To enable software practitioners to perform data exploration and analysis in order to obtain insightful and actionable information for data-driven tasks around software.

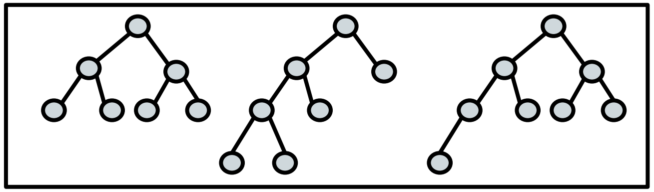
Defect prediction

Linkable questions prediction

software analytics



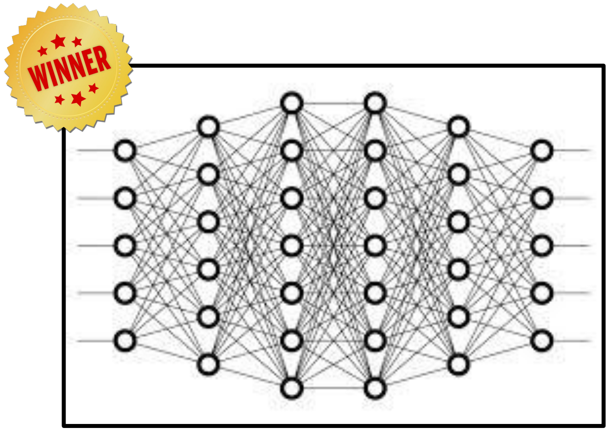


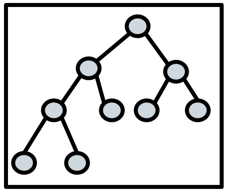


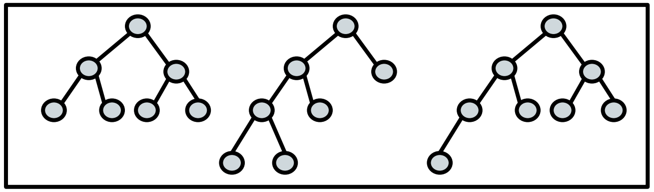
Defect prediction

Linkable questions prediction

software analytics



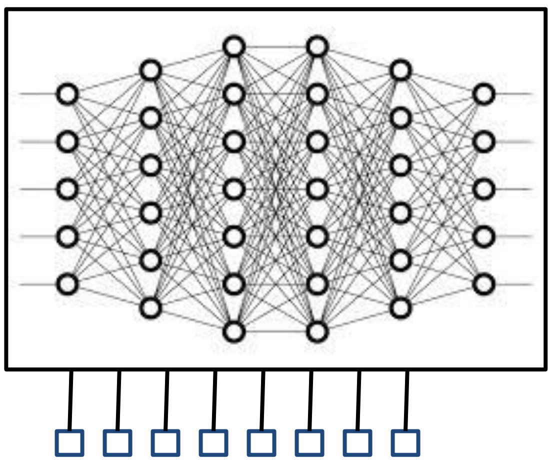


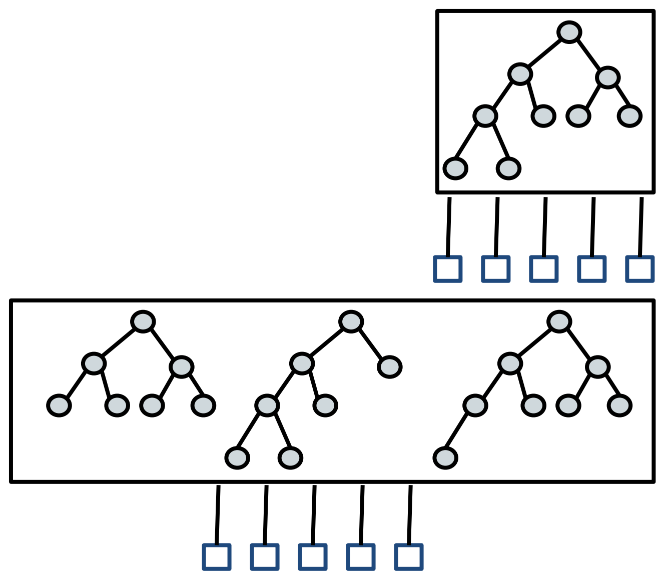


Defect prediction

Linkable questions prediction

software analytics

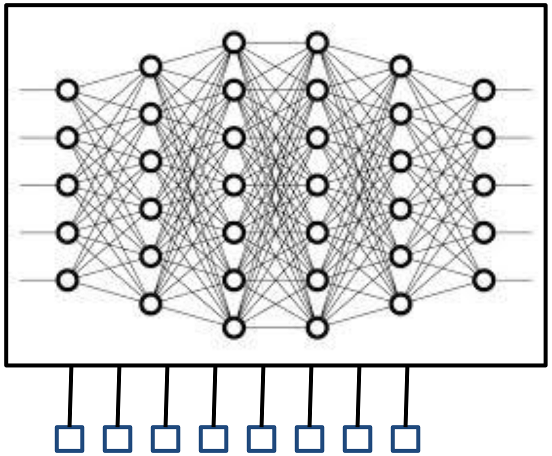


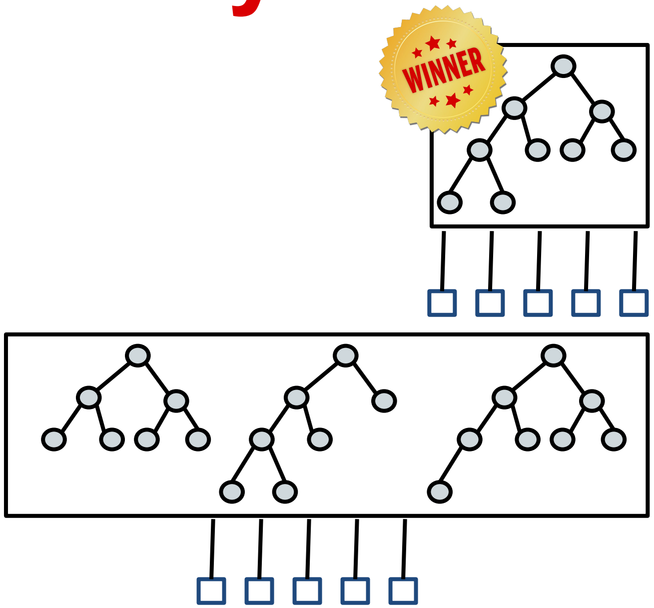


Defect prediction

Linkable questions prediction

software analytics



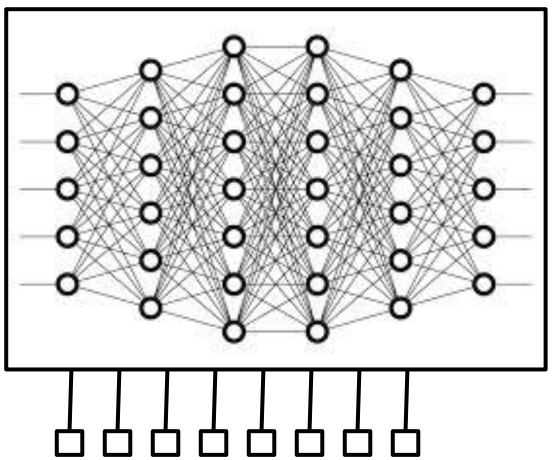


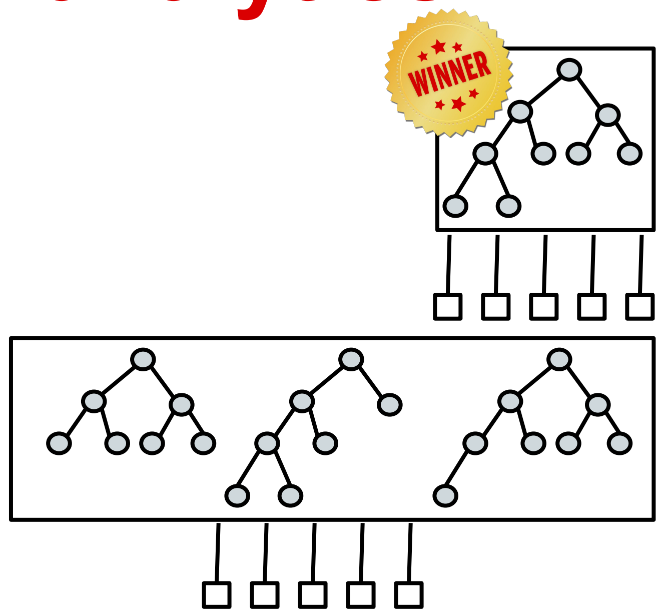
Defect prediction

Linkable questions prediction

TUNING!

software analytics



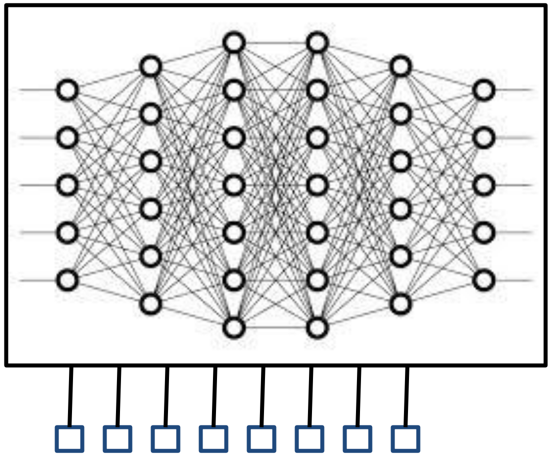


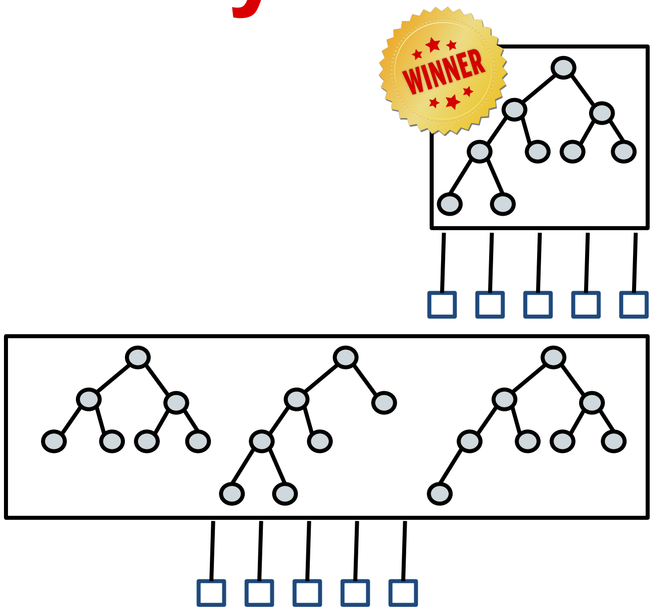
Defect prediction

Linkable questions prediction

TUNING (with DE)!

software analytics





- ***Is tuning with DE <u>helpful</u>?***

- ***Is tuning with DE a <u>faster</u> method?***

- ***How to <u>improve</u> tuning with DE?***

- ***Is tuning with DE <u>helpful</u>?***
  - Tuning for defect predictors (**IST**'16)
  - Tuning for topic modeling (**IST**, minor revision)

- ***Is tuning with DE a <u>faster</u> method?***

- ***How to <u>improve</u> tuning with DE?***

- ***Is tuning with DE <u>helpful</u>?***
  - Tuning for defect predictors (**IST**'16)
  - Tuning for topic modeling (**IST**, minor revision)

- ***Is tuning with DE a <u>faster</u> method?***
  - DE v.s. grid search (under review)
  - DE+SVM v.s. deep learning (**FSE**'17)

- ***How to <u>improve</u> tuning with DE?***

Heterogeneous Defect Prediction **(TSE'17)**

JIT Effort-aware Defect Prediction **(FSE'17)**

- ***Is tuning with DE <u>helpful</u>?***
  - Tuning for defect predictors (**IST**'16)
  - Tuning for topic modeling (**IST**, minor revision)

- ***Is tuning with DE a <u>faster</u> method?***
  - DE v.s. grid search (under review)
  - DE+SVM v.s. deep learning (**FSE**'17)

- ***How to <u>improve</u> tuning with DE?***
  - **Future work...**

***Tuning* for every task**

Knowledge reuse

Tuning should be faster

***Simple* method first**

Faster methods for software analytics!

Why Faster Software Analytics?

- CPU
- Cost (cloud service)
- Reproducibility

- Arcuri et al[Arcuri2011] reported their tuning require ***weeks, or more***, of CPU time.
- Wang et al[wang2013] require ***weeks to years*** to learn control settings.
- Deep learning:
  - Lam et al.[Lam2015]: ***weeks*** of CPU.
  - Gu et al.[Gu2016]: ***240 hours*** of GPU.

Why Faster Software Analytics?

- CPU
- Cost (cloud service)
- Reproducibility

AWS cost: Computing + Bandwidth
+ Storage +...

Why Faster Software Analytics?

- CPU
- Cost (cloud service)
- Reproducibility

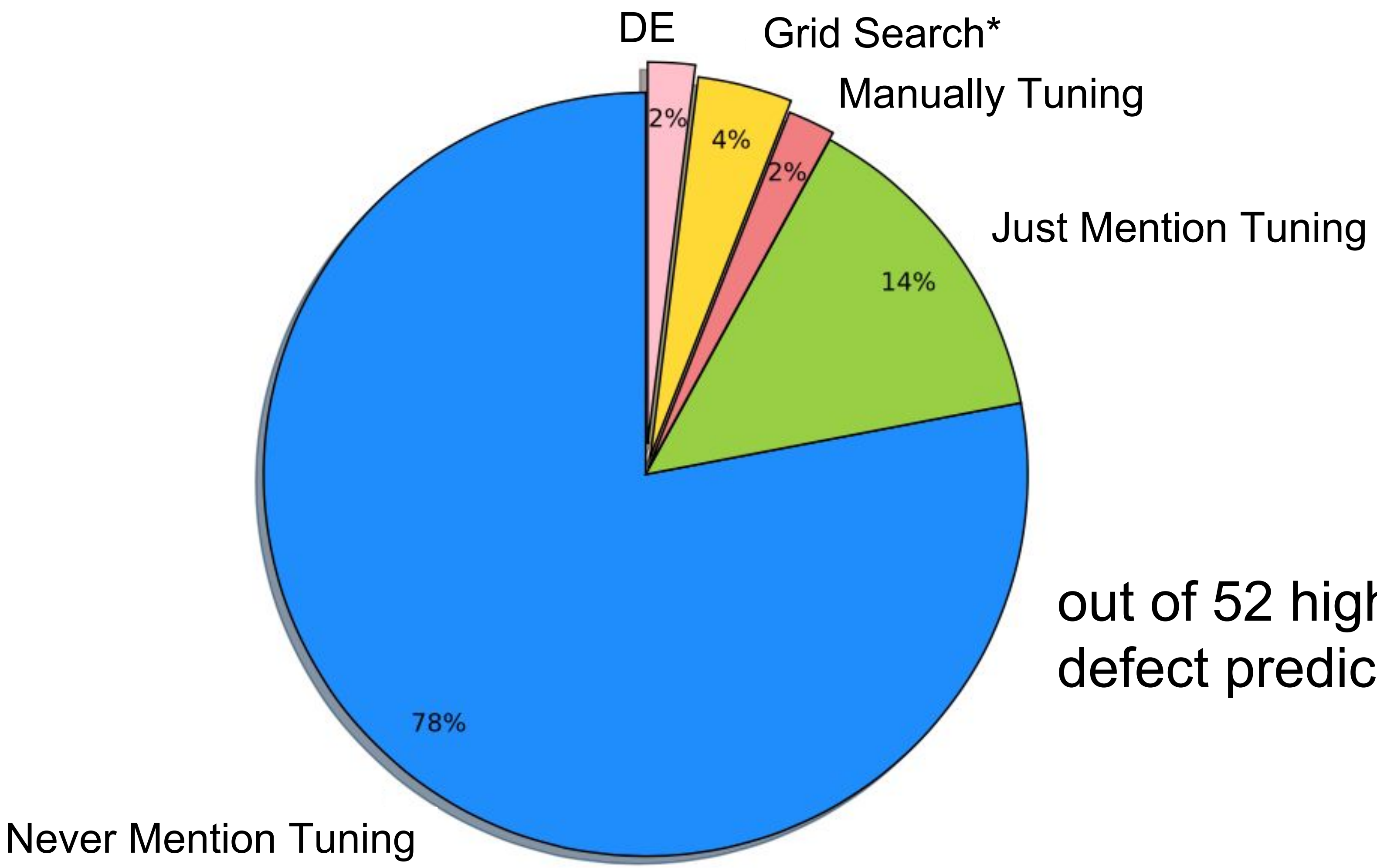
Wang et al[Wang 2013] ***15 years*** of CPU time to do code clone detection

TUNING (with DE)!

Tuning is <u>Ignored</u> in SE!

* Bergstra, James, and Yoshua Bengio. "Random search for hyper-parameter optimization." *Journal of Machine Learning Research* 13.Feb (2012): 281-305.

Tuning is Ignored in SE!



out of 52 highly cited defect prediction papers

* Bergstra, James, and Yoshua Bengio. "Random search for hyper-parameter optimization." *Journal of Machine Learning Research* 13.Feb (2012): 281-305.

Why Tuning Ignored?

Cause they are so well explored all already… right?

CPU intensive!

- ***Is tuning with DE helpful?***
    - Tuning for defect predictors (**IST**’16)
    - Tuning for topic modeling (**IST**, minor revision)
- ***Is tuning with DE a faster method?***
    - DE v.s. grid search (under review)
    - DE+SVM v.s. deep learning (**FSE**’17)
- ***How to improve tuning with DE?***
    - **Future work...**

This talk FSE'17

Easy over Hard: A Case Study on Deep Learning

Wei Fu, Tim Menzies
Com.Sci., NC State, USA
wfu@ncsu.edu,tim.menzies@gmail.com

ABSTRACT

While deep learning is an exciting new technique, the benefits of this method need to be assessed with respect to its computational cost. This is particularly important for deep learning since these learners need hours (to weeks) to train the model. Such long training time limits the ability of (a) a researcher to test the stability of their conclusion via repeated runs with different random seeds; and (b) other researchers to repeat, improve, or even refute that original work.

For example, recently, deep learning was used to find which questions in the Stack Overflow programmer discussion forum can be linked together. That deep learning system took 14 hours to execute. We show here that applying a very simple optimizer called DE to fine tune SVM, it can achieve similar (and sometimes better)

semantically related, they are considered as *linkable* knowledge units.

In their paper, they used a convolution neural network (CNN), a kind of deep learning method [42], to predict whether two KUs are linkable. Such CNNs are highly computationally expensive, often requiring network composed of 10 to 20 layers, hundreds of millions of weights and billions of connections between units [42]. Even with advanced hardware and algorithm parallelization, training deep learning models still requires hours to weeks. For example:

- XU report that their analysis required 14 hours of CPU.
- Le [40] used a cluster with 1,000 machines (16,000 cores) for three days to train a deep learner.

This paper debates what methods should be recommended to those wishing to repeat the analysis of XU. We focus on whether

DE + SVM ✓ ***Deep learning*** ✗

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 ?


Method

Case Study

Linkable Questions Prediction on StackOverflow

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

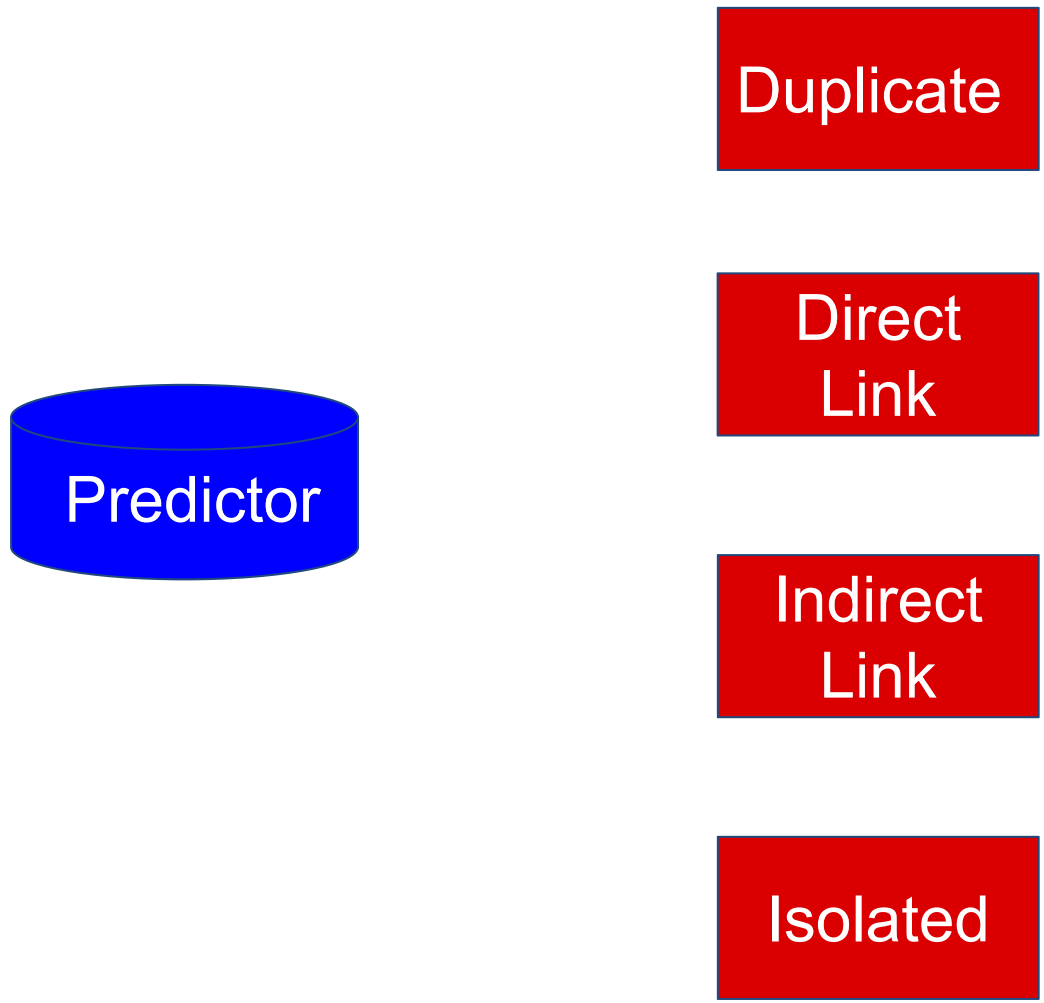
ASE'16

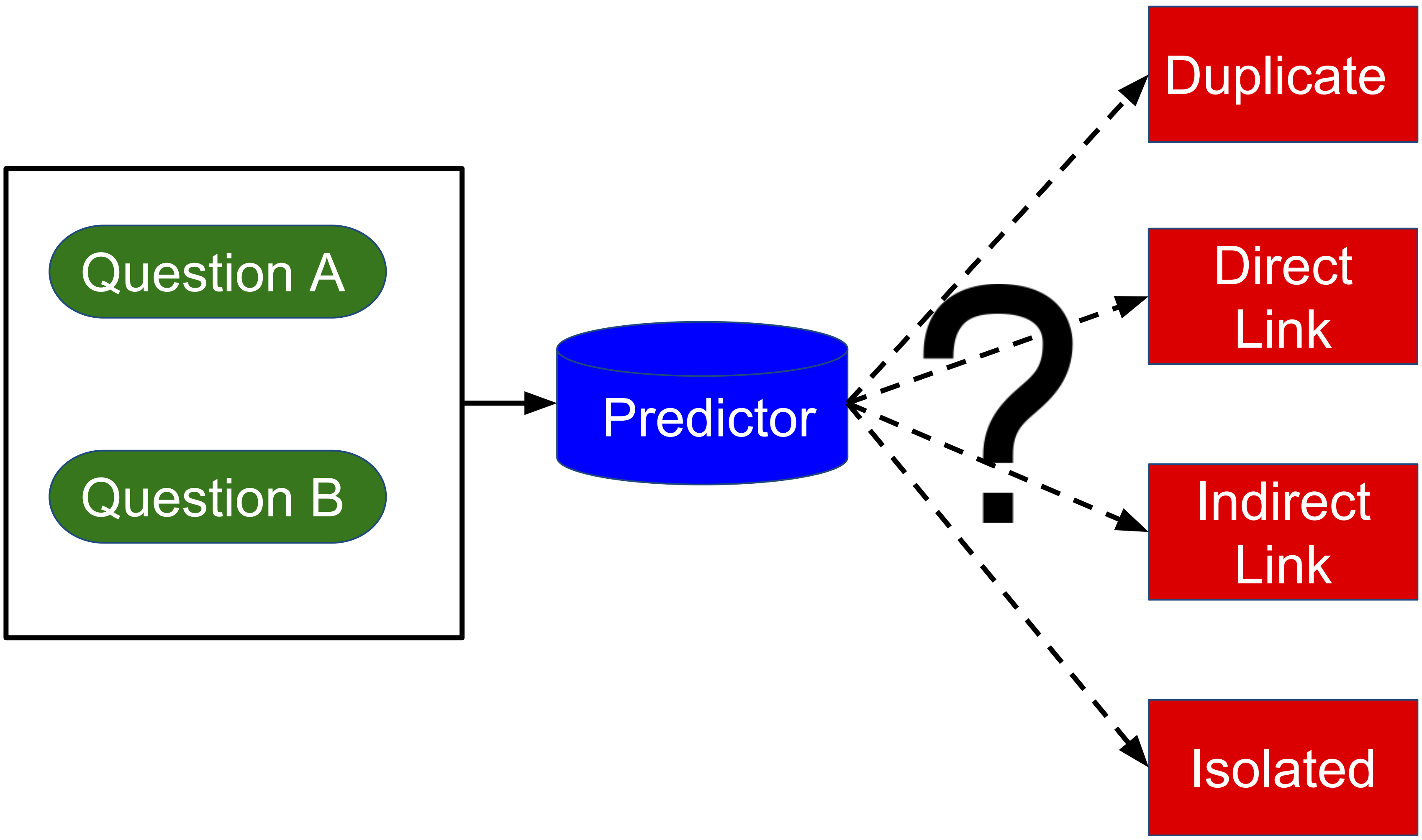
Duplicate

Direct Link

Indirect Link

Isolated


Predictor
Duplicate
Direct Link
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***

Parameters in SVM (scikit-learn):
C, kernel, gamma, coef0

Tuning Algorithm: Differential Evolution*

Population = Pick N options at random # e.g. N =10

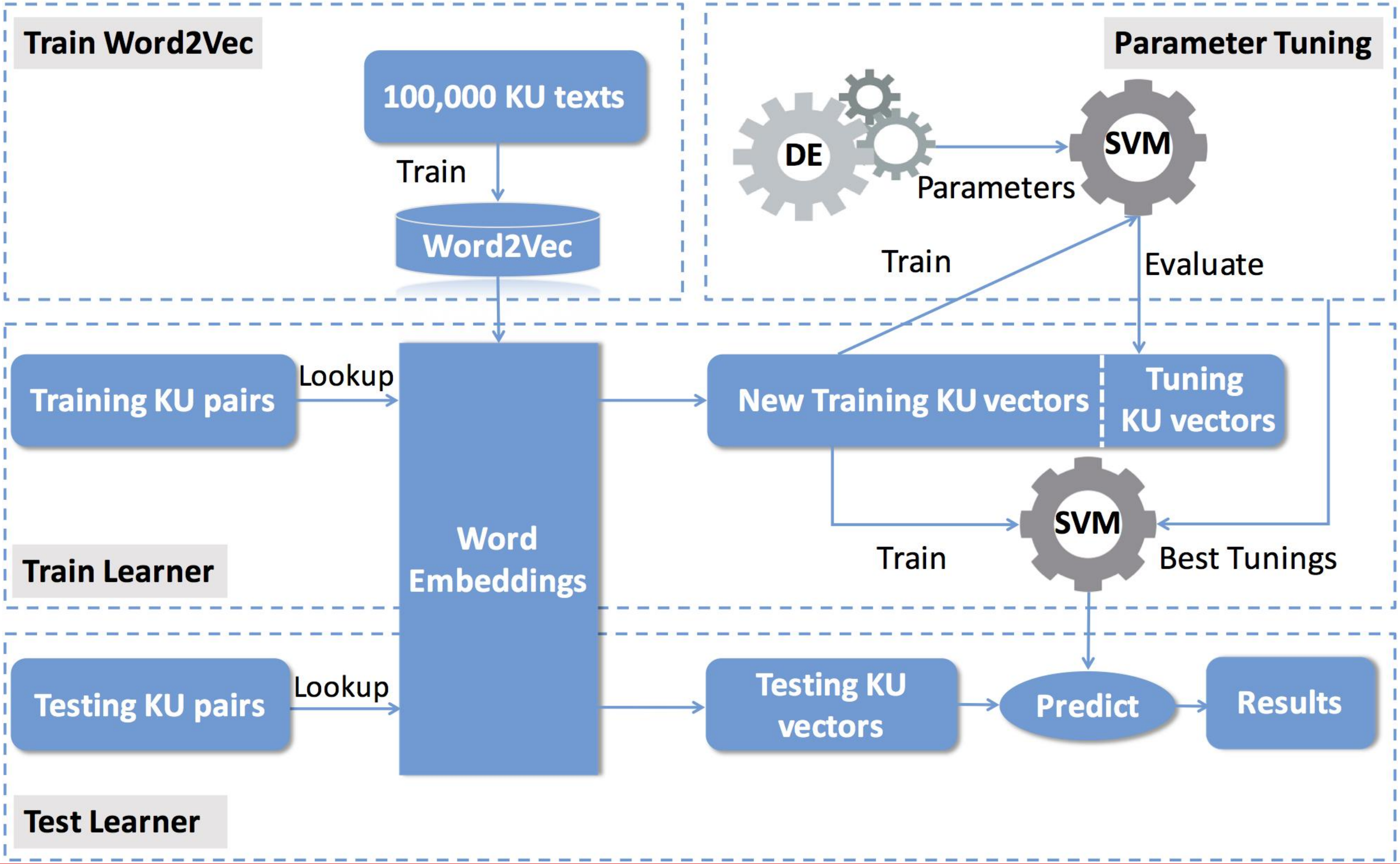
M times repeat : # e.g. M = 5

for Parent in Population:

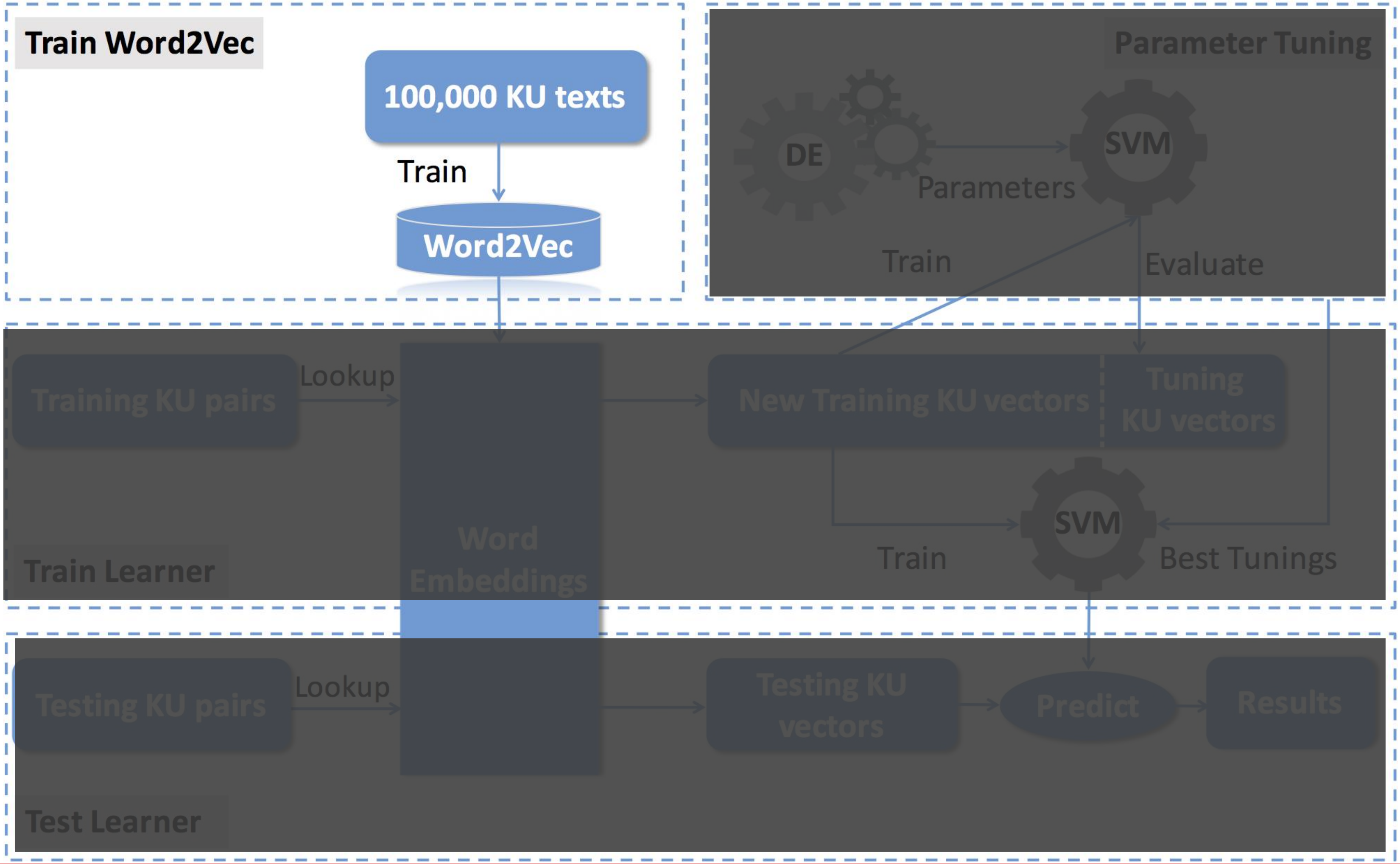
- Select a, b, c = three other items in population.
- Candidate = a + f*(b-c) # ish
- if Candidate “better”, replace Parent.

* 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.

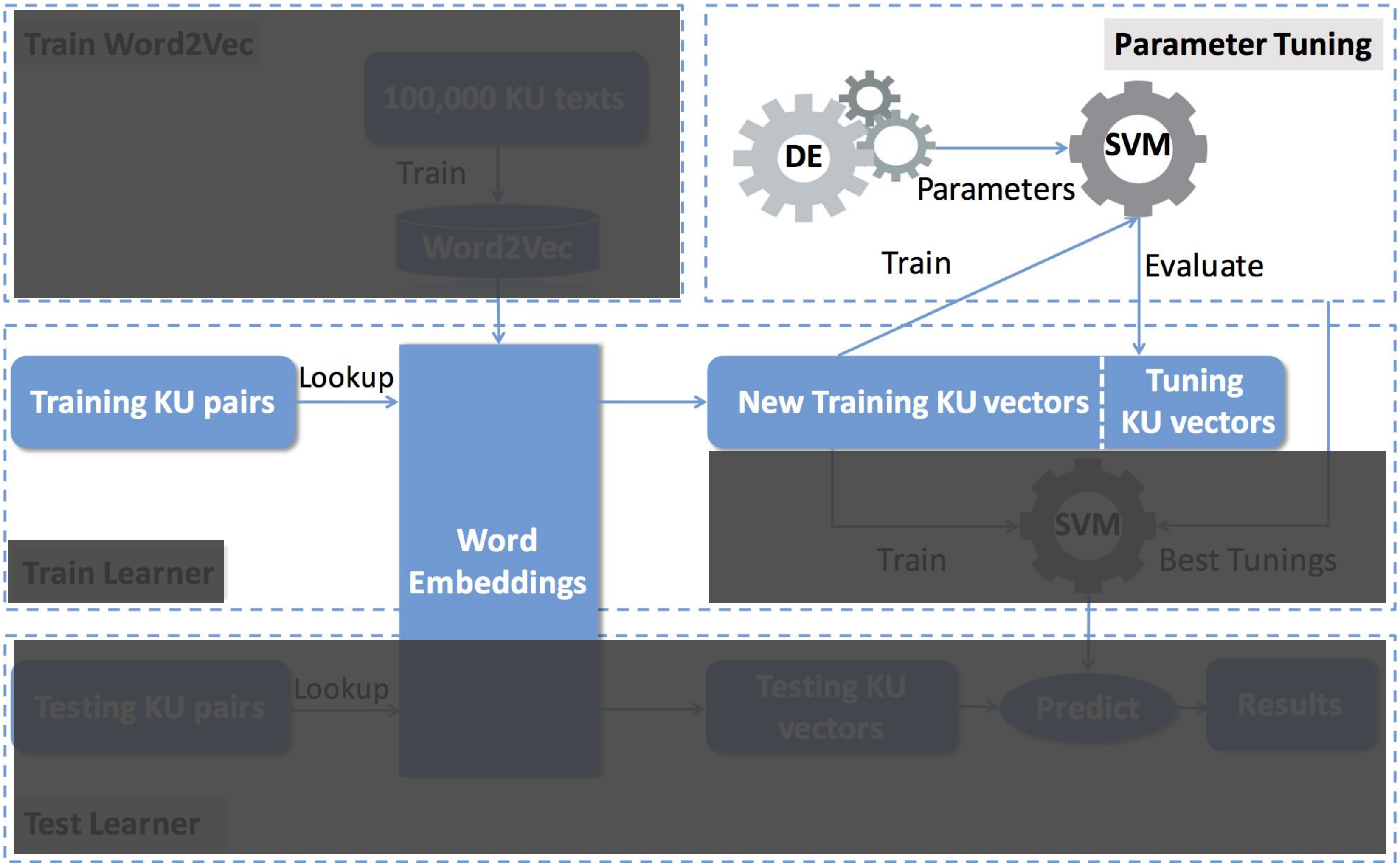
Experimental Setup



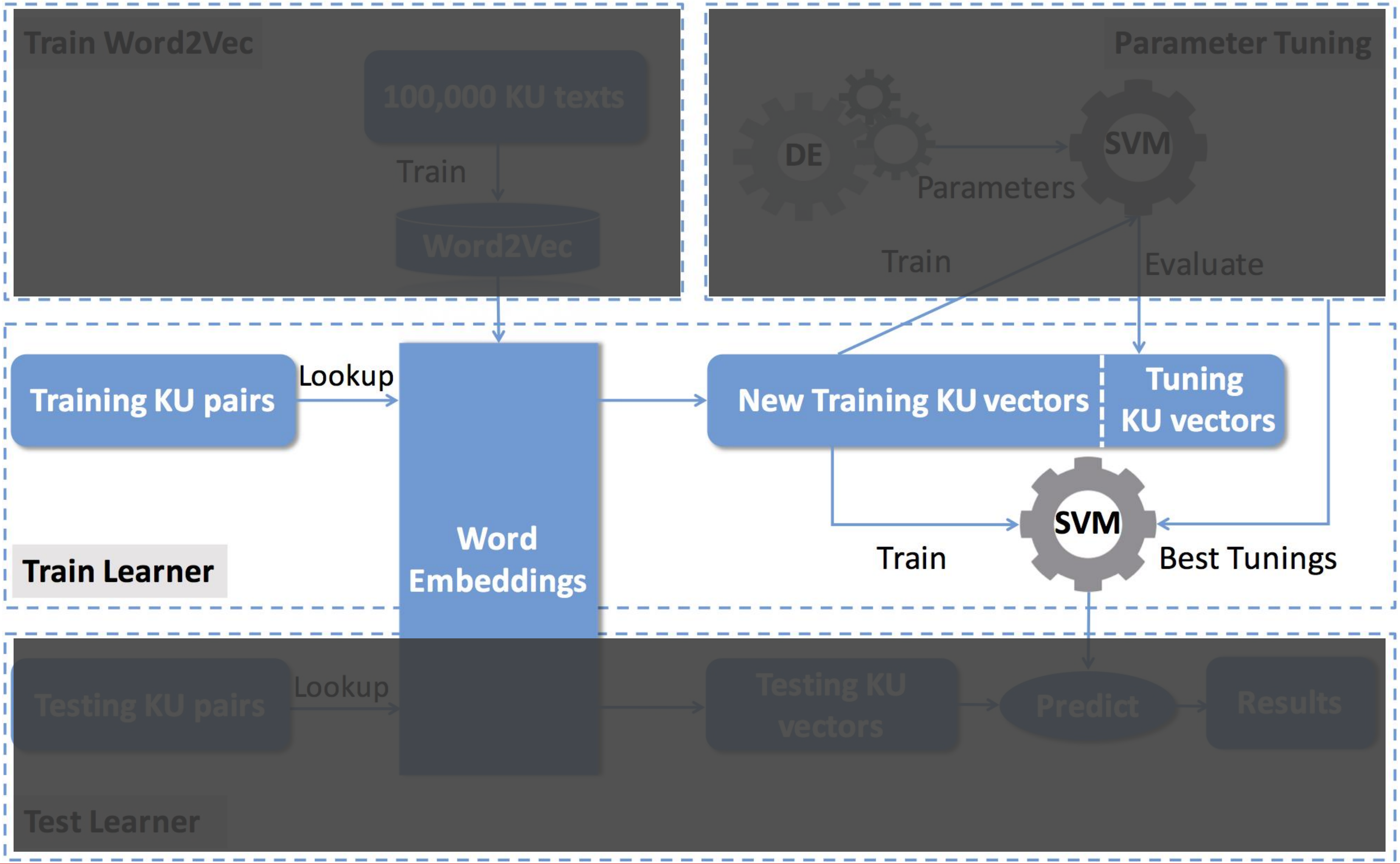
Experimental Setup



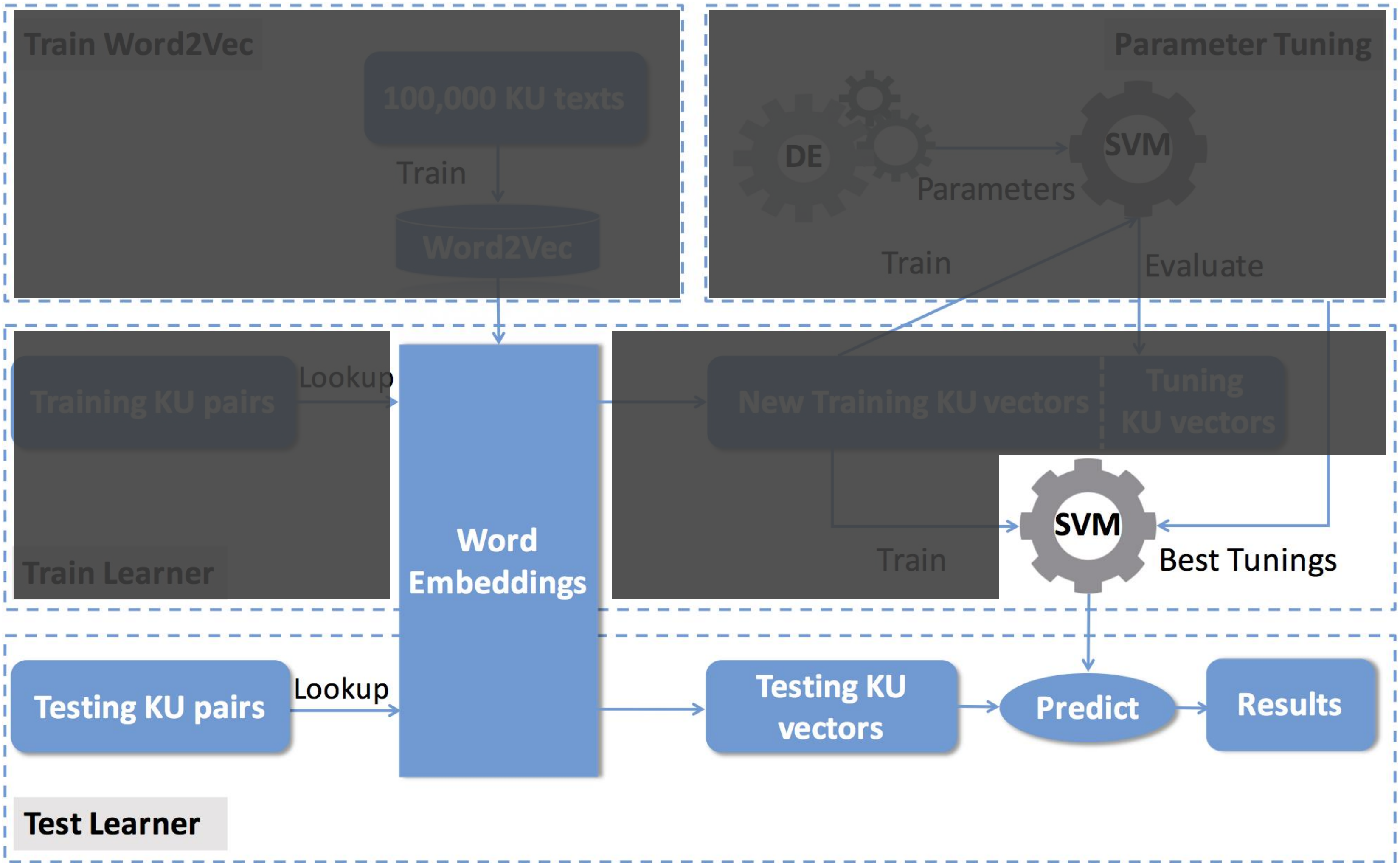
Experimental Setup



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	**0.72**	0.51	0.77	0.60	0.65
	XU's SVM	0.61	**0.56**	**0.78**	**0.67**	**0.65**
Recall	Our SVM	0.52	**0.49**	0.97	**0.64**	0.65
	XU's SVM	**0.72**	0.43	**0.98**	0.53	**0.66**
F1-score	Our SVM	0.60	**0.50**	0.86	**0.62**	0.65
	XU's SVM	**0.66**	0.48	**0.87**	0.60	**0.65**

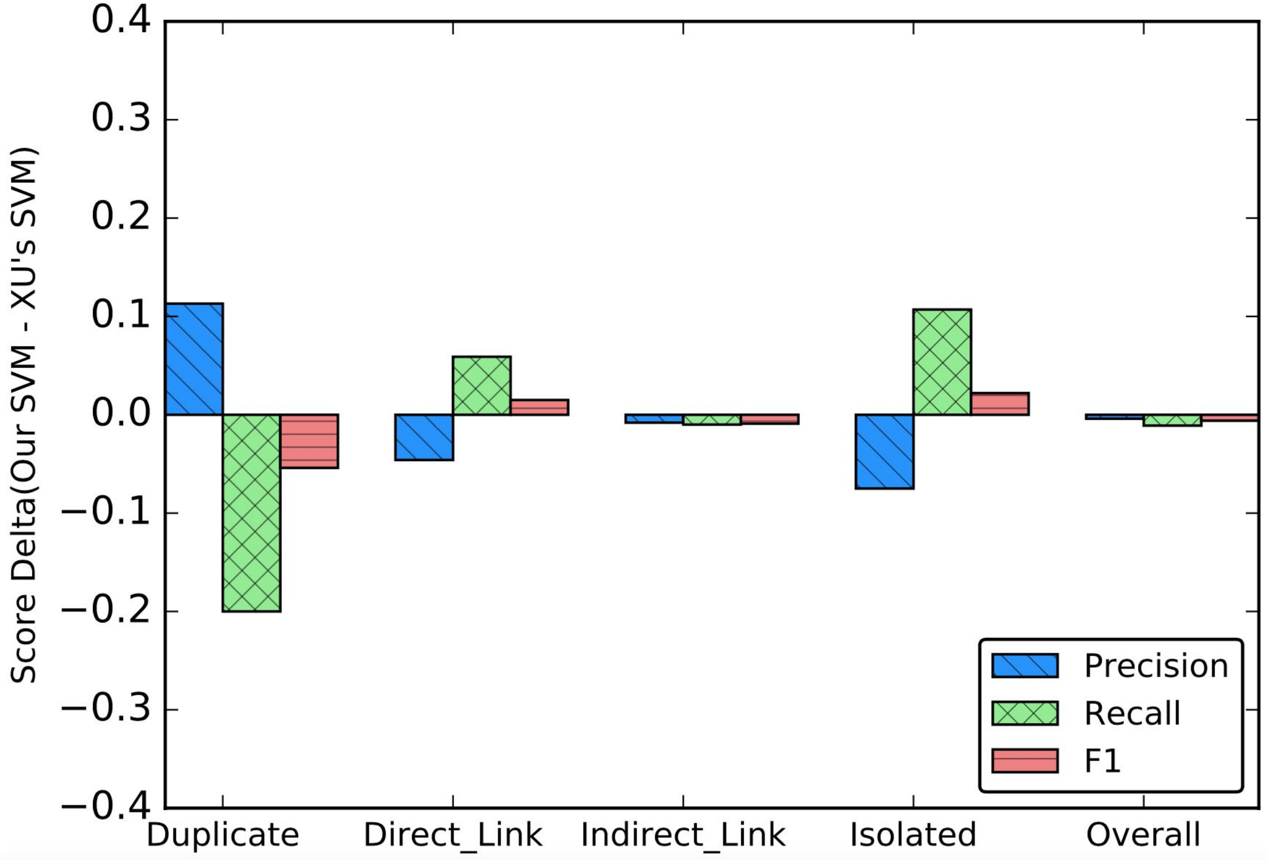
RQ1: Reproduce Xu's Baseline Results?

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

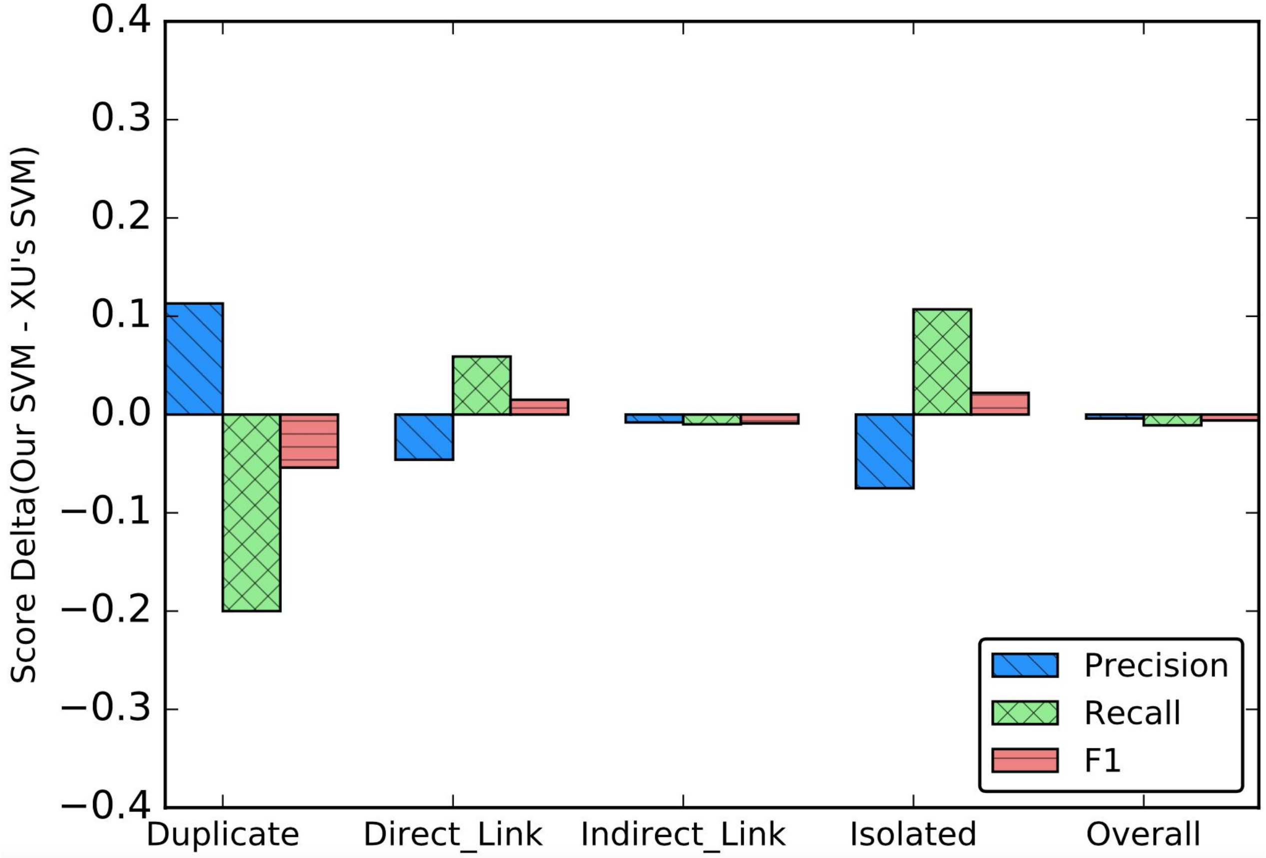
Metrics	Methods	Duplicate	Direct Link	Indirect Link	Isolated	Overall
Precision	Our SVM	**0.72**	0.51	0.77	0.60	0.65
	XU's SVM	0.61	**0.56**	**0.78**	**0.67**	**0.65**
Recall	Our SVM	0.52	**0.49**	0.97	**0.64**	0.65
	XU's SVM	**0.72**	0.43	**0.98**	0.53	**0.66**
F1-score	Our SVM	$\boxed{0.60}$	**0.50**	0.86	**0.62**	0.65
	XU's SVM	$\boxed{\mathbf{0.66}}$	0.48	**0.87**	0.60	**0.65**

Score Delta(F1) = Our SVM - Xu's SVM = -0.06

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

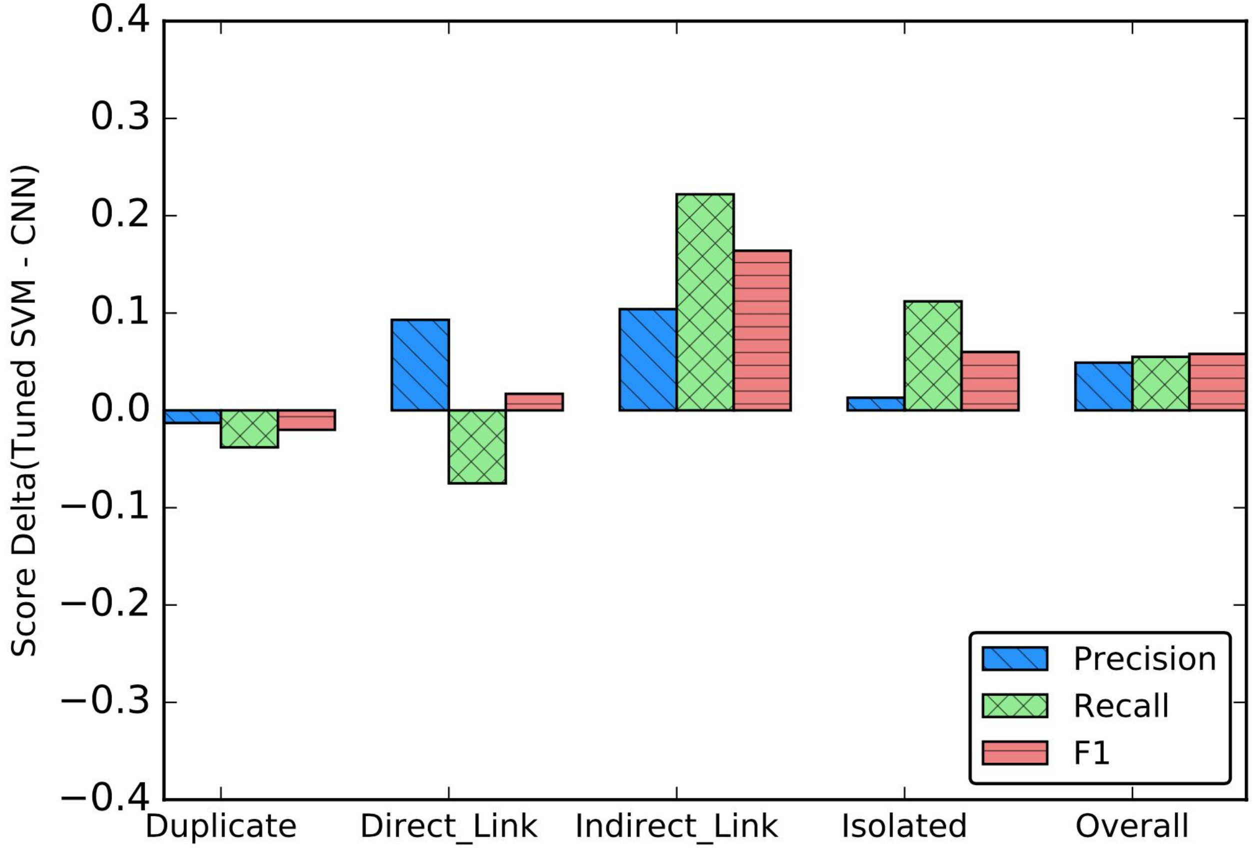
Research Questions

RQ1: Can we reproduce Xu’s baseline results?

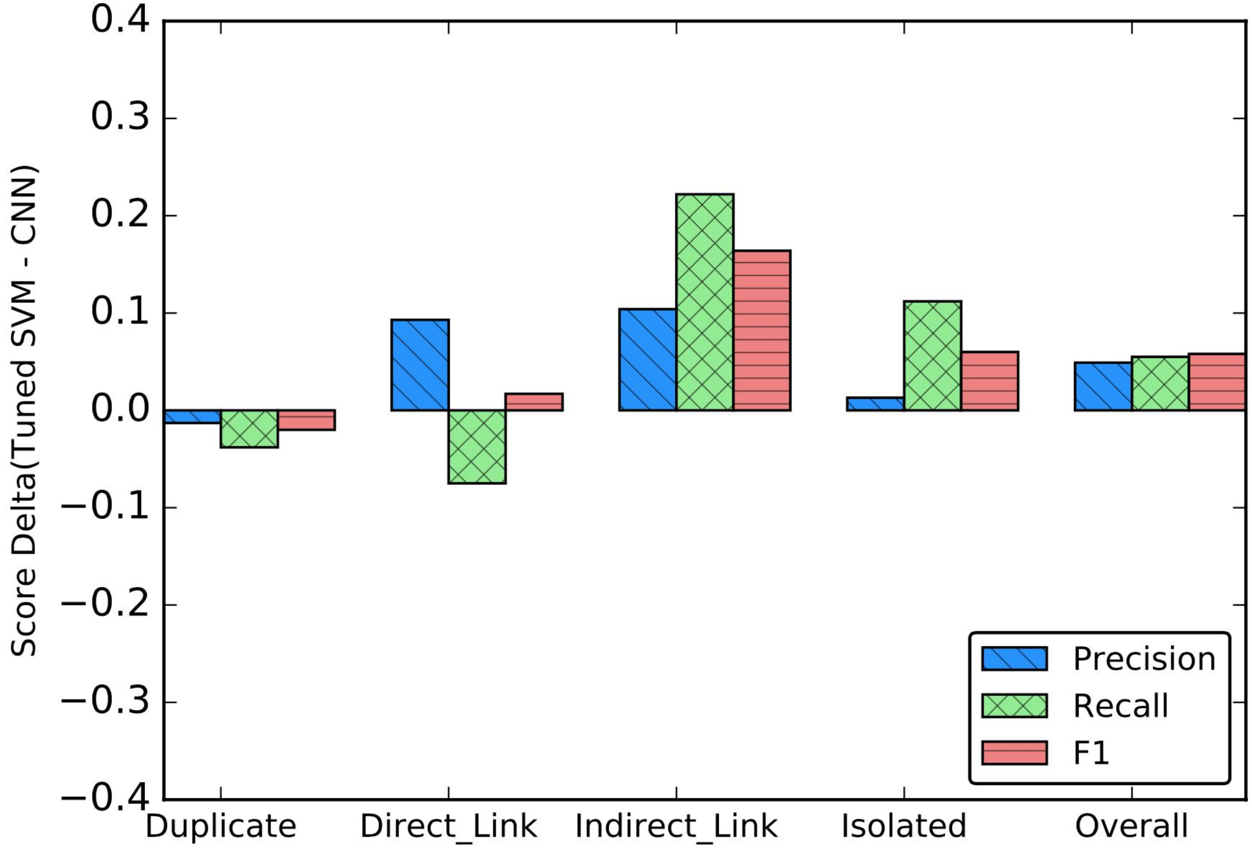
RQ2: DE+SVM outperforms Xu’s deep learning method?

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.

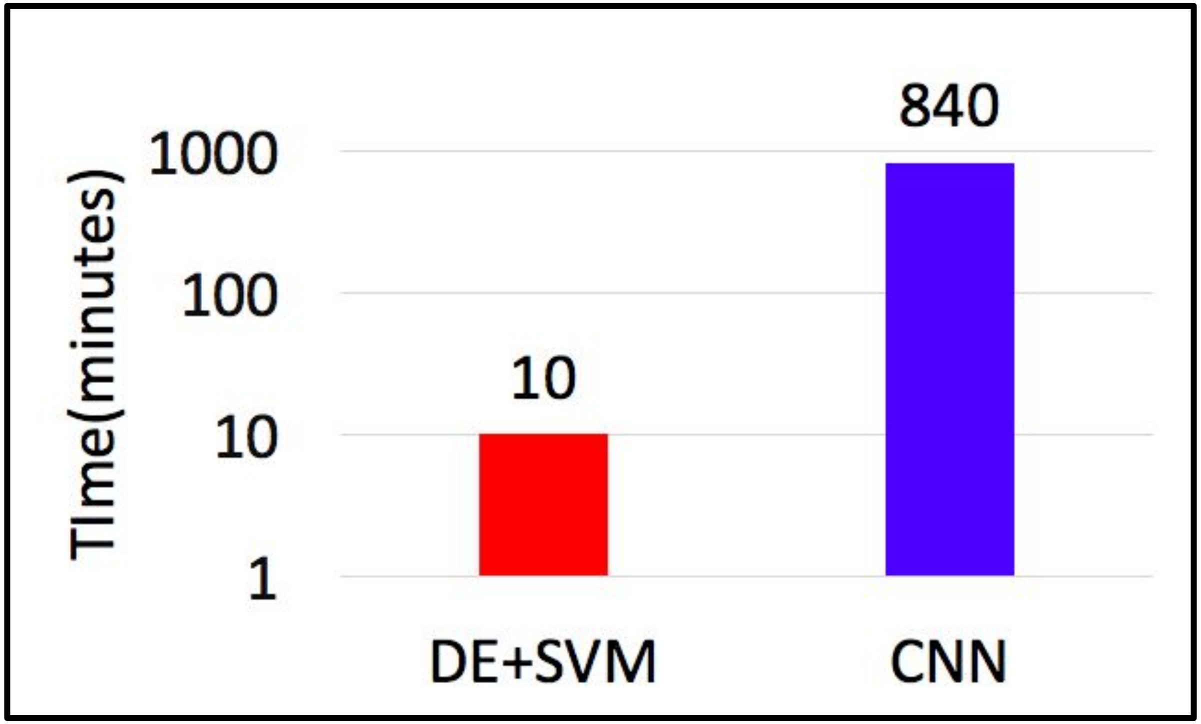
Research Questions

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

For this case study:

Simple DE tuning performs
Better & Faster than deep learning!

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.

Implication

For future deep learning in SE:

- **TUNE** your baseline methods.
- Do not ignore the **COST** of deep learning.

- ***Is tuning with DE <u>helpful</u>?***
  - Tuning for defect predictors (**IST**'16)
  - Tuning for topic modeling (**IST**, minor revision)

- ***Is tuning with DE a <u>faster</u> method?***
  - DE v.s. grid search (under review)
  - <u>DE+SVM v.s. deep learning (**FSE**'17)</u>

- ***How to <u>improve</u> tuning with DE?***
  - **Future work...**

10 minutes tuning is NOT TRUE for all SE tasks!

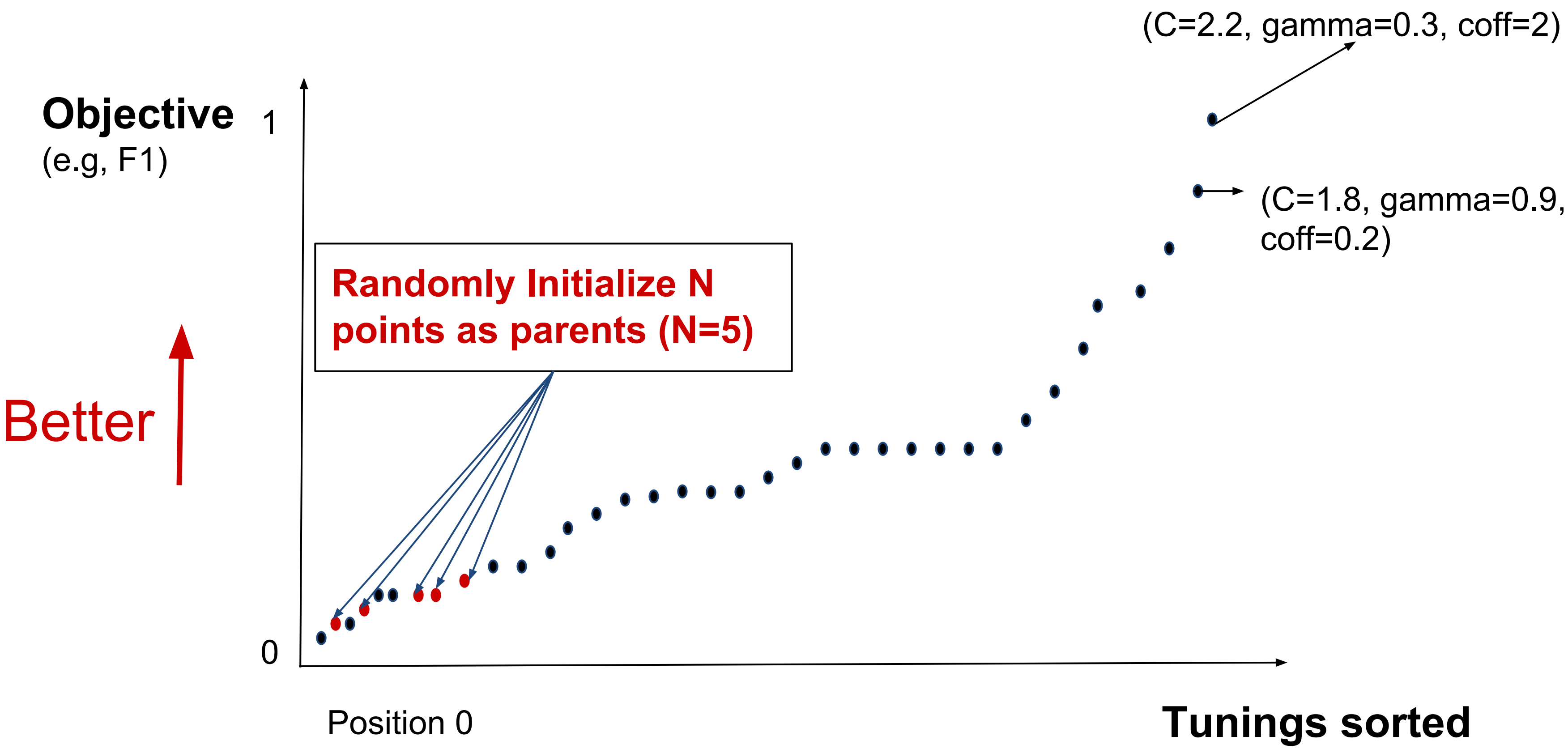
That depends on:

- Learners (SVM, random forests, deep learning,...)
- Software analytic tasks (data, goal,….)
- Searching algorithms (DE, GA,....)

Challenge

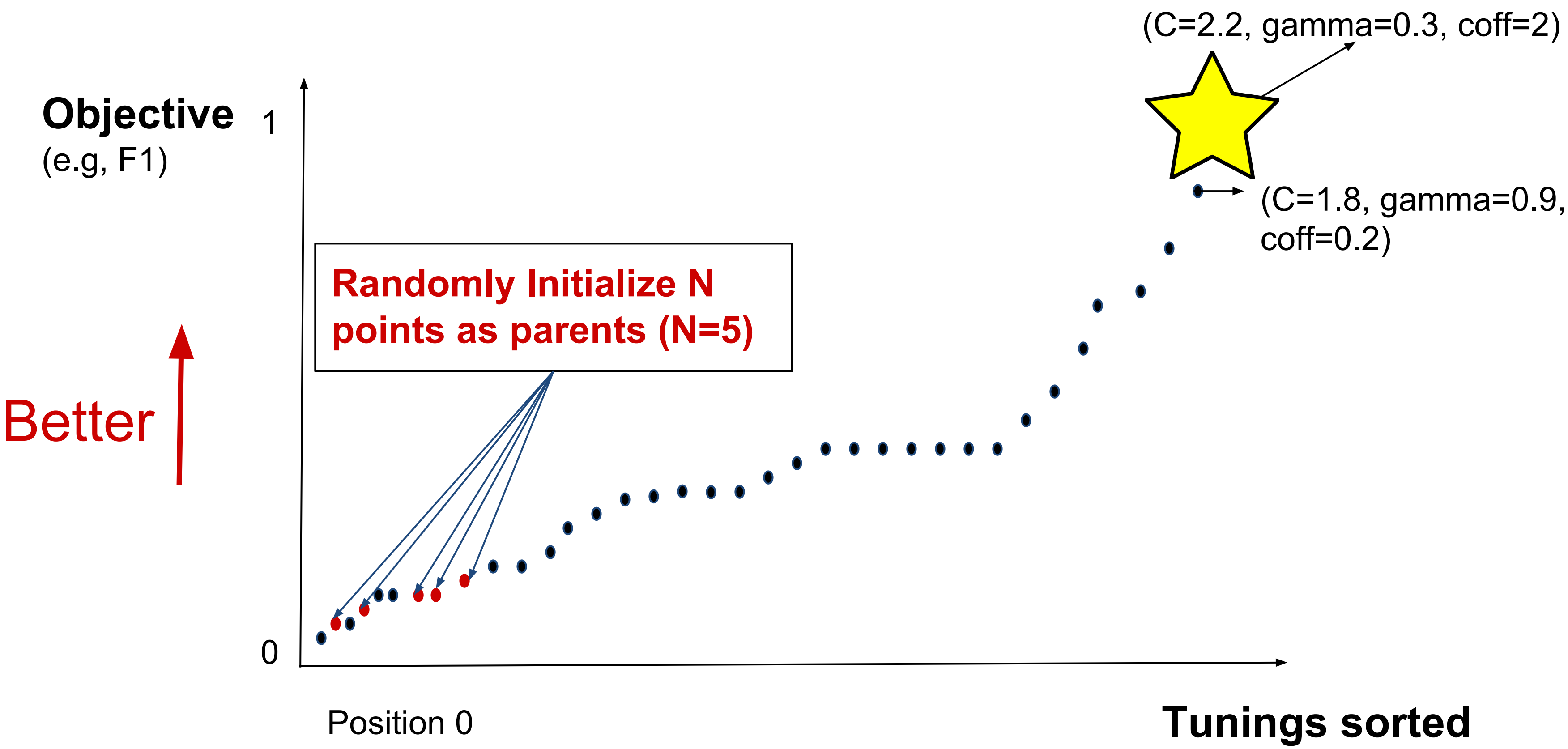
Given a limited budget, can we improve performance of tuning?

Recap on Tuning with DE



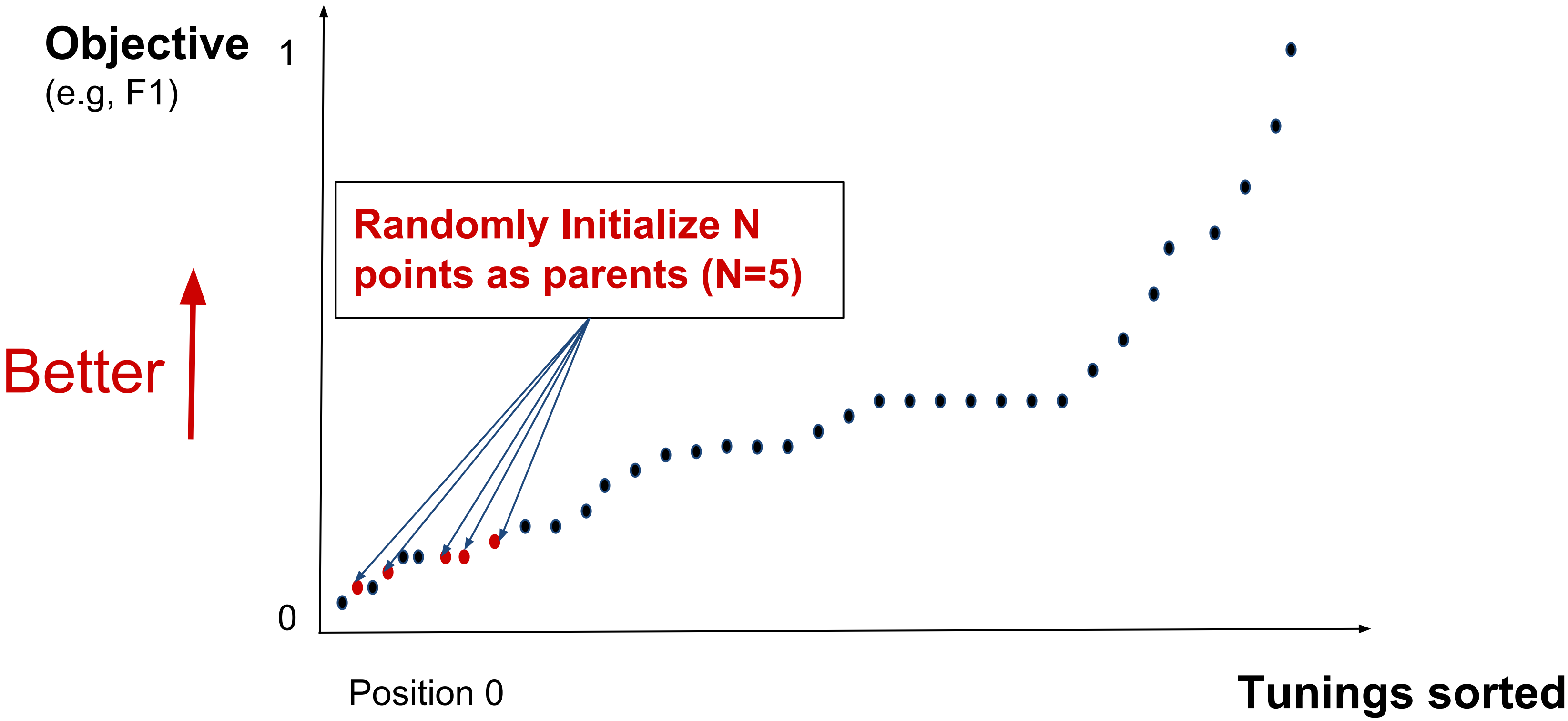
Objective space, points represent scores of tunings (parameters), e.g. F1 score of (C=1.2, gamma=0.5, coff=1)= 0.3

Recap on Tuning with DE



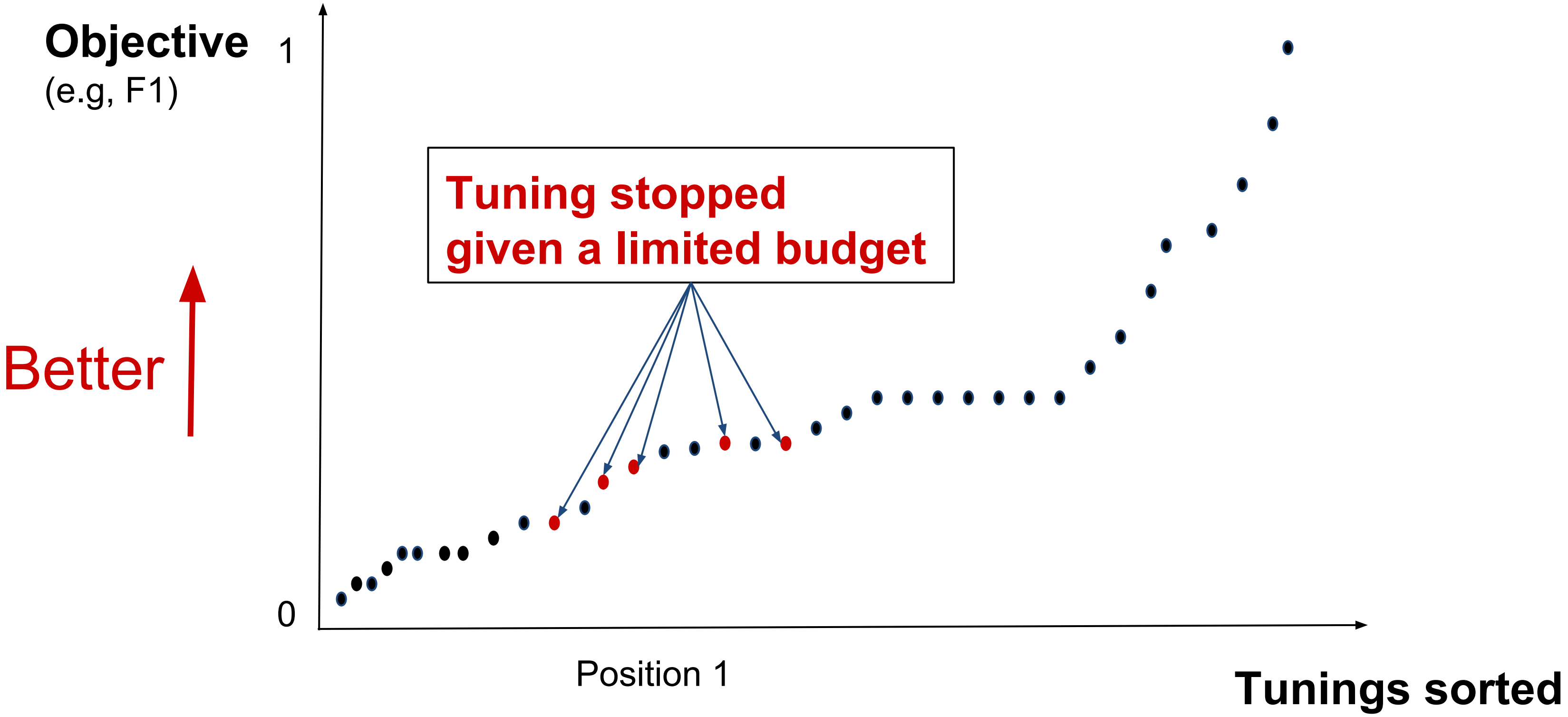
Objective space, points represent scores of tunings (parameters), e.g. F1 score of (C=1.2, gamma=0.5, coff=1)= 0.3

Recap on Tuning with DE



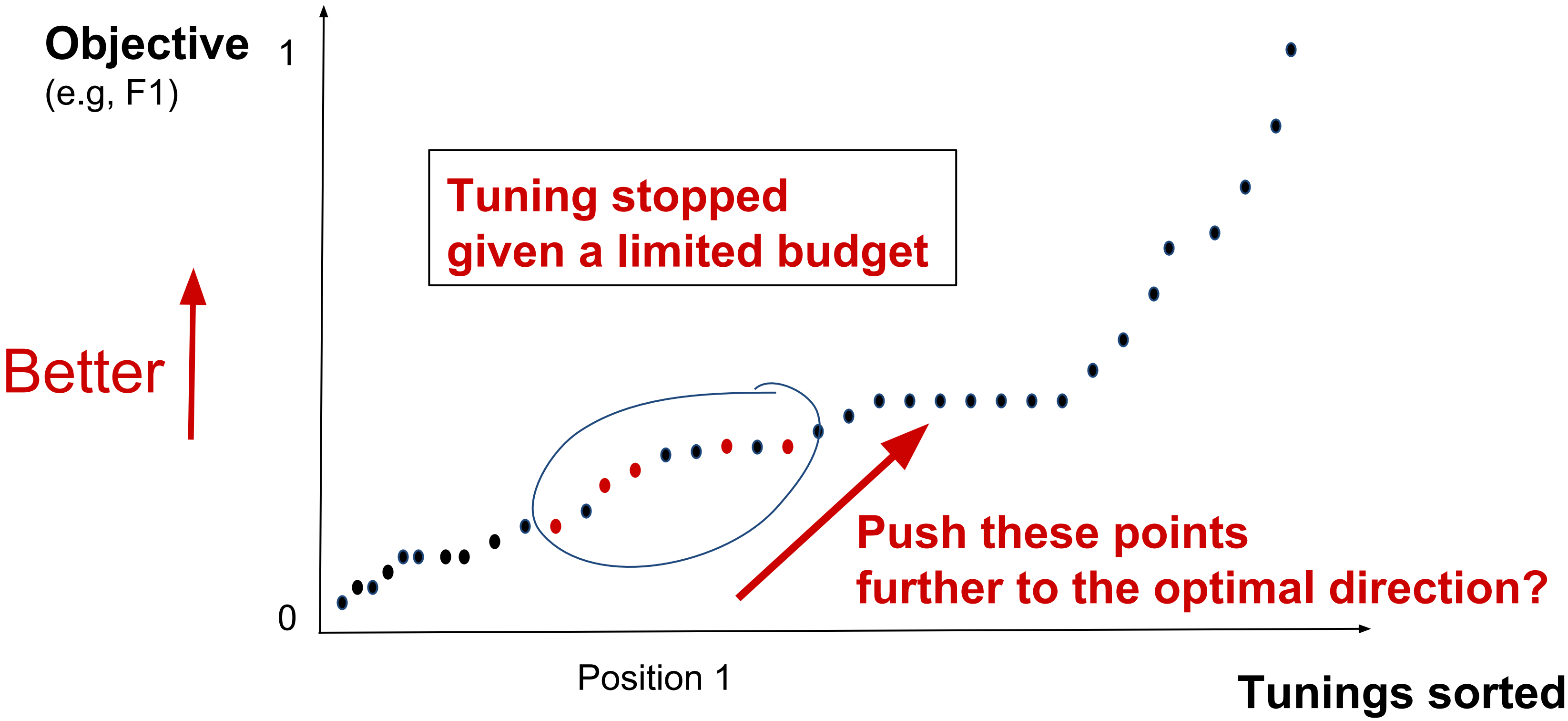
Objective space, points represent scores of tunings (parameters), e.g. F1 score of (C=1.2, gamma=0.5, coff=1)= 0.3

Recap on Tuning with DE



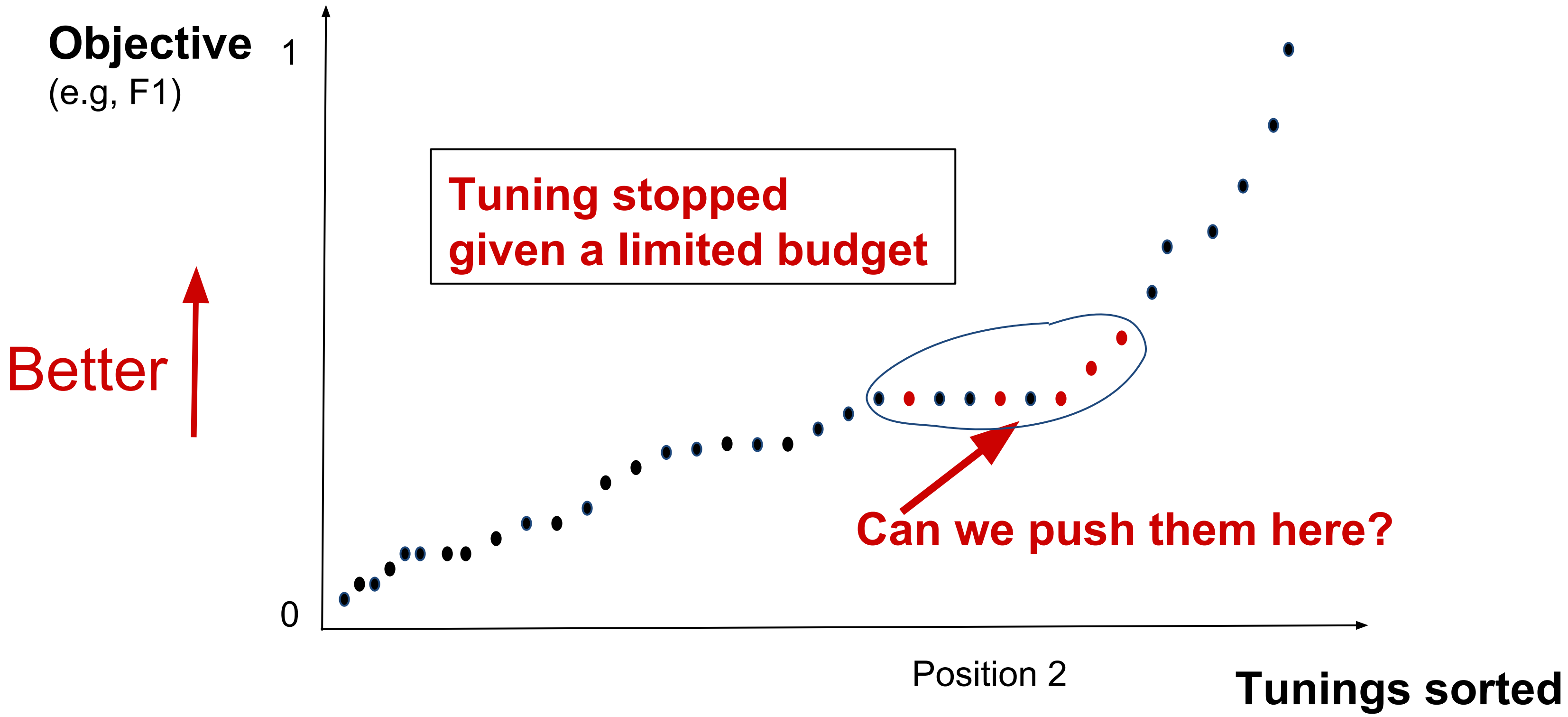
Objective space, points represent scores of tunings (parameters).

Recap on Tuning with DE



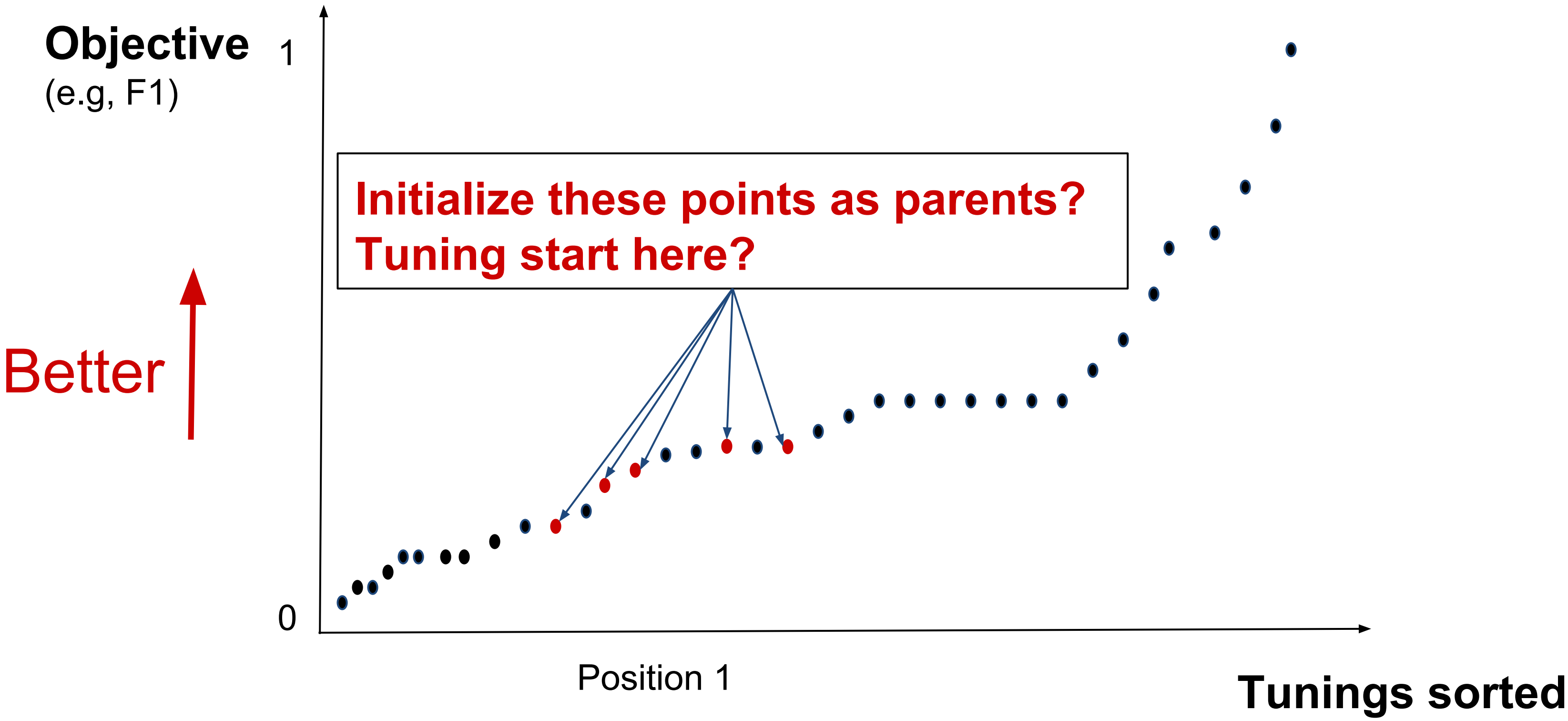
Objective space, points represent scores of tunings (parameters).

Recap on Tuning with DE



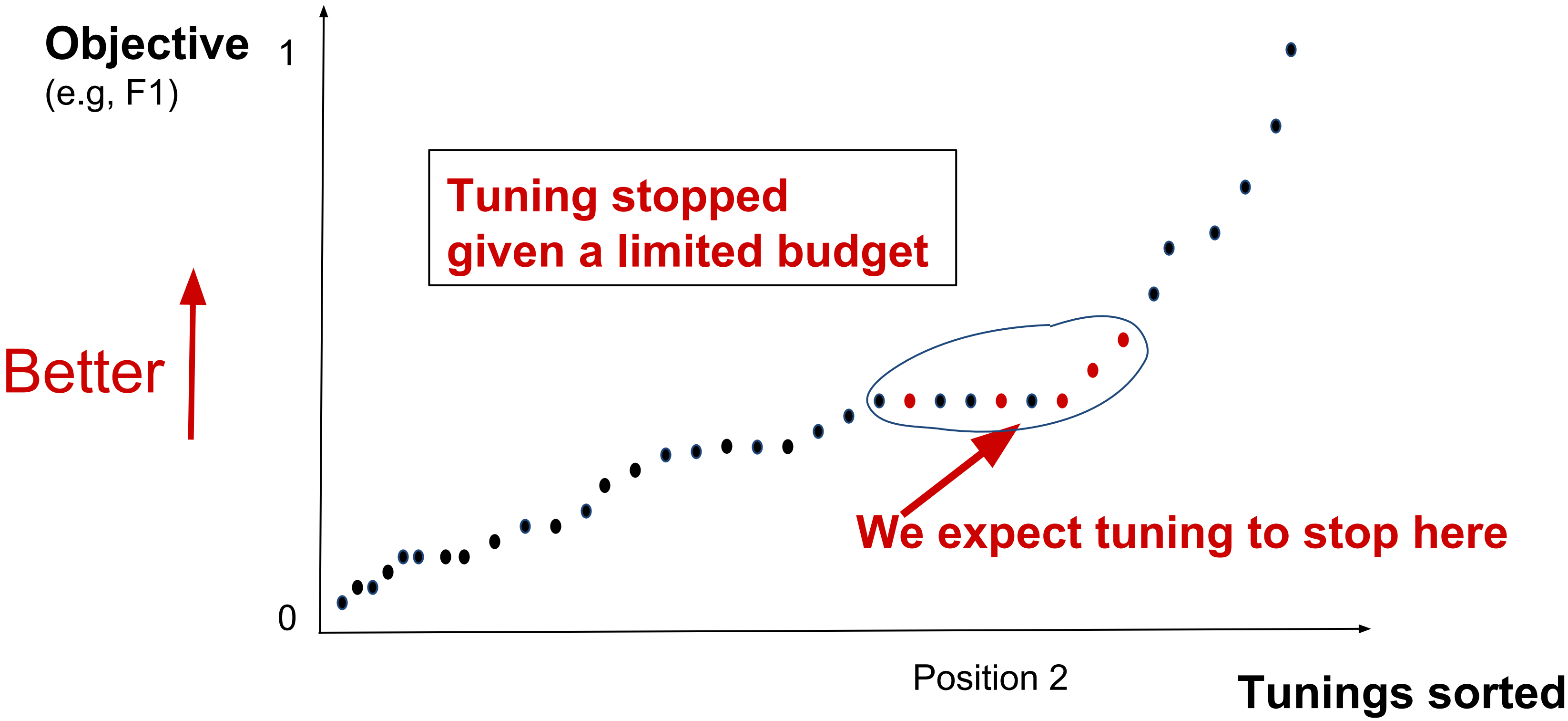
Objective space, points represent scores of tunings (parameters).

Tuning with DE: Better Initialization?



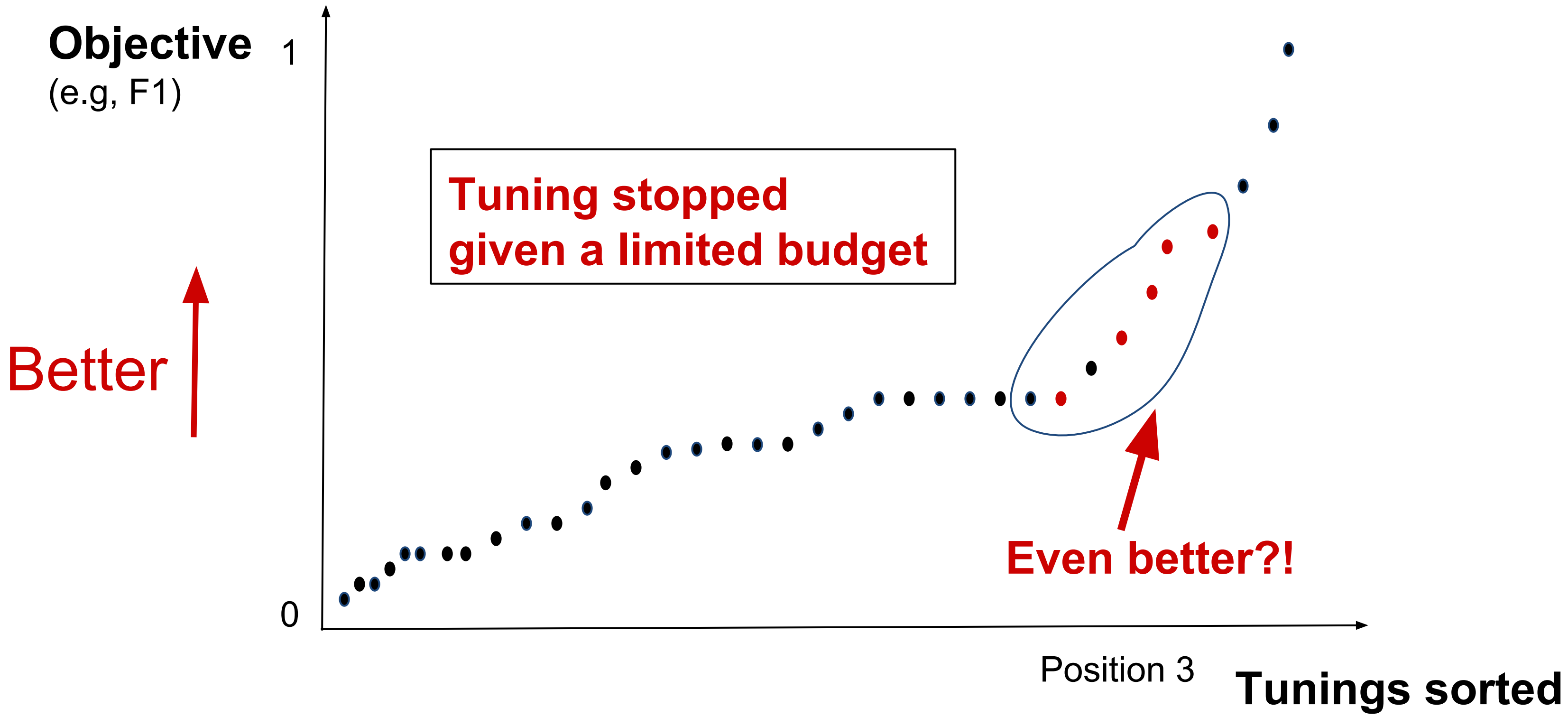
Objective space, points represent scores of tunings (parameters).

Tuning with DE: Get Better Results?



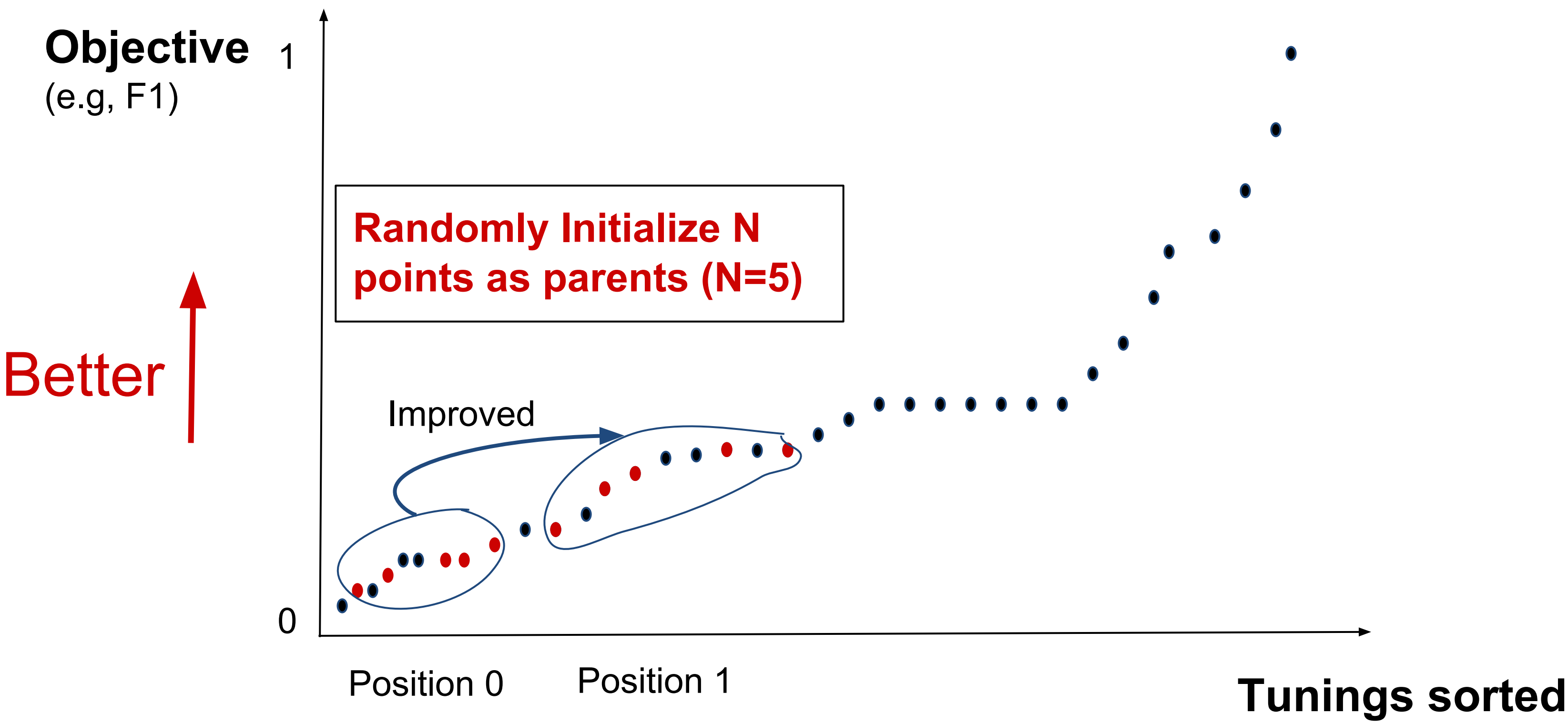
Objective space, points represent scores of tunings (parameters).

Tuning with DE: Get Even Better?



Objective space, points represent scores of tunings (parameters).

We Need a Better Initialization!



Objective space, points represent scores of tunings (parameters).

We Got Some Experience...

IEEE TRANS SE. SUBMITTED JAN'16, REVISION#?, APR'16 1

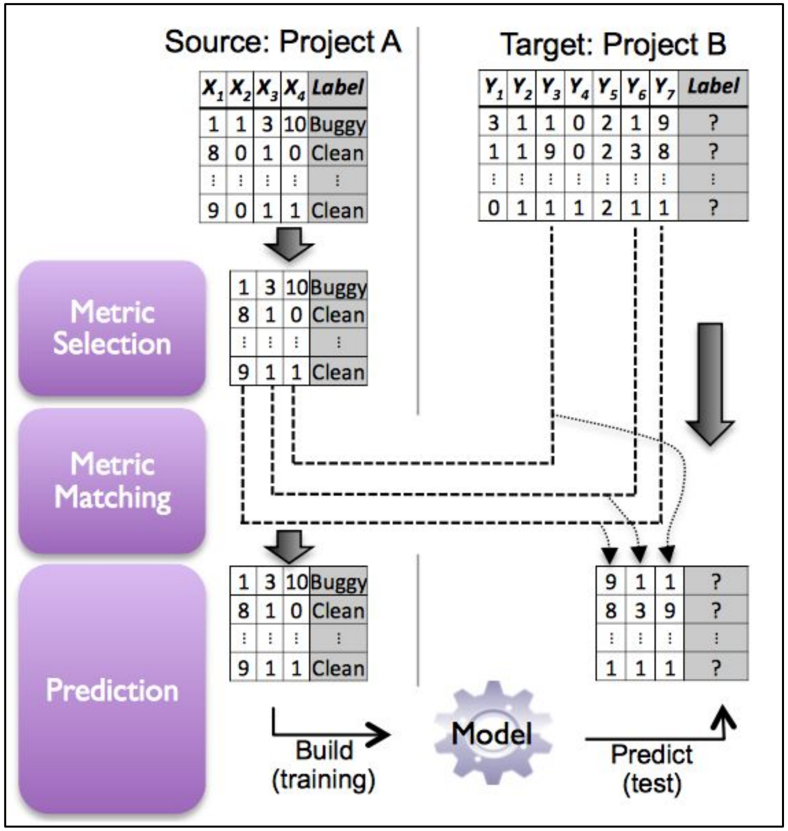
Heterogeneous Defect Prediction

Jaechang Nam, Wei Fu, Sunghun Kim, *Member, IEEE*,
Tim Menzies, *Member, IEEE*, and Lin Tan, *Member, IEEE*

Abstract—Many recent studies have documented the success of cross-project defect prediction (CPDP) to predict defects for new projects lacking in defect data by using prediction models built by other projects. However, most studies share the same limitations: it requires homogeneous data; i.e., different projects must describe themselves using the *same* metrics. This paper presents methods for *heterogeneous* defect prediction (HDP) that matches up different metrics in different projects. Metric matching for HDP requires a "large enough" sample of distributions in the source and target projects—which raises the question on how large is "large enough" for effective heterogeneous defect prediction. This paper shows that empirically and theoretically, "large enough" may be very small indeed. For example, using a mathematical model of defect prediction, we identify categories of data sets were as few as 50 instances are enough to build a defect prediction model. Our conclusion for this work is that, even when projects use different metric sets, it is possible to quickly transfer lessons learned about defect prediction.

Index Terms—defect prediction, quality assurance, heterogeneous metrics, transfer learning.

TSE 2017



Other Researchers Reported...

SEAMS 2017

2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)

Transfer Learning for Improving Model Predictions in Highly Configurable Software

Pooyan Jamshidi, Miguel Velez, Christian Kästner
Carnegie Mellon University, USA
{pjamshid,mvelezce,kaestner}@cs.cmu.edu

Norbert Siegmund
Bauhaus-University Weimar, Germany
norbert.siegmund@uni-weimar.de

Prasad Kawthekar
Stanford University, USA
pkawthek@stanford.edu

ICPE 2017

Transferring Performance Prediction Models Across Different Hardware Platforms

Pavel Valov
University of Waterloo
200 University Avenue West
Waterloo, ON, Canada
pvalov@uwaterloo.ca

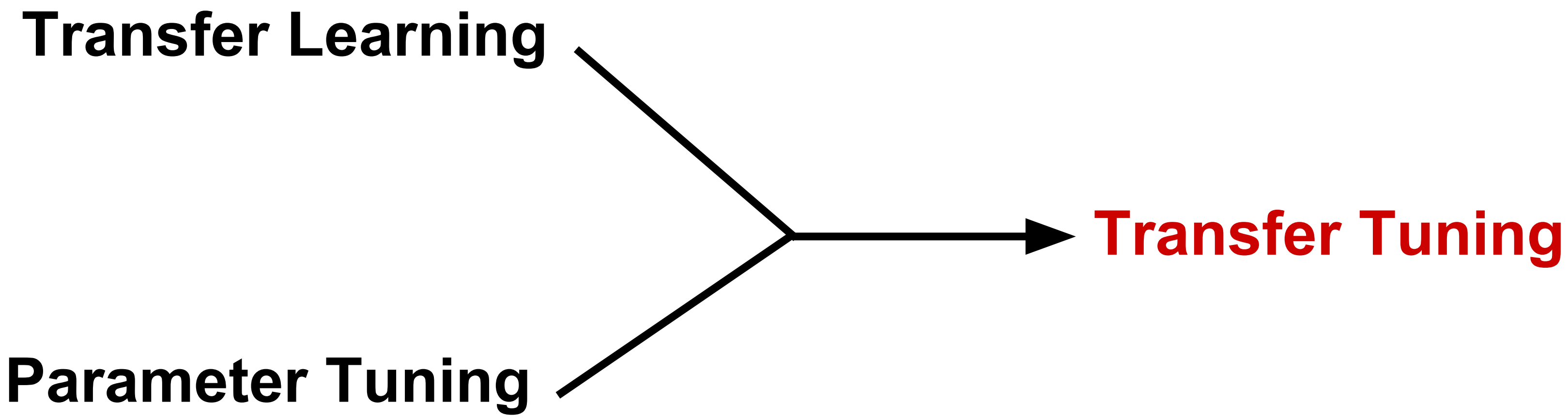
Jean-Christophe Petkovich
University of Waterloo
200 University Avenue West
Waterloo, ON, Canada
j2petkovich@uwaterloo.ca

Jianmei Guo*
East China University of Science and Technology
130 Meilong Road
Shanghai, China
gjm@ecust.edu.cn

Sebastian Fischmeister
University of Waterloo
200 University Avenue West
Waterloo, ON, Canada
sfischme@uwaterloo.ca

Krzysztof Czarnecki*
University of Waterloo
200 University Avenue West
Waterloo, ON, Canada
kczarnec@gsd.uwaterloo.ca

Our Proposed Idea



Another Illustration

Differential Evolution:

Population = 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.

Another Illustration

Present:

projectN → Population = Pick N options **at random** # e.g. N =10

.

.

.

project3 → Population = Pick N options **at random** # e.g. N =10

project2 → Population = Pick N options **at random** # e.g. N =10

project1 → Population = Pick N options **at random** # e.g. N =10

Another Illustration

Future:

projectN

Population = Pick N options # e.g. N =10

.

.

.

project3

Population = Pick N options # e.g. N =10

project2

Population = Pick N options # e.g. N =10

project1

Population = Pick N options **at random** # e.g. N =10

Transfer Knowledge

Preliminary Results - Defect Prediction

Learner: CART, Performance metric: Precision

CamelV0

- No transfer ==> 0.521
- Transfer from log4j ==> 0.667
- Transfer from luence ==> 0.667

CamelV1

- No transfer ==> 0.398
- Transfer from jeditV2 ==> 0.8
- Transfer from log4j ==> 0.8
- Transfer from poiV1 ==> 0.8

Preliminary Results - Defect Prediction

Learner: CART, Performance metric: F1

XercesV1

- No Transfer ==> 0.399
- Transfer from poiV1 ==> 0.56
- Transfer from synapse ==> 0.488

PoiV0

- No Transfer ==> 0.728
- Transfer from antV2 ==> 0.804
- Transfer from synapse ==> 0.819

Challenges

- Better transfer learning strategy for transfer tuning.
- Understand why and when transfer learning works for tuning.
- How to generalize to other software analytics?

My progress so far....

Parameter Tuning

- Apr,16 Tuning (IST 2016)
- Dec,16 DE better? (Under Review)
- Jan,17 DE + LDA[1] (IST Minor Revision)
- Jun,17 Easy over Hard(FSE'17)

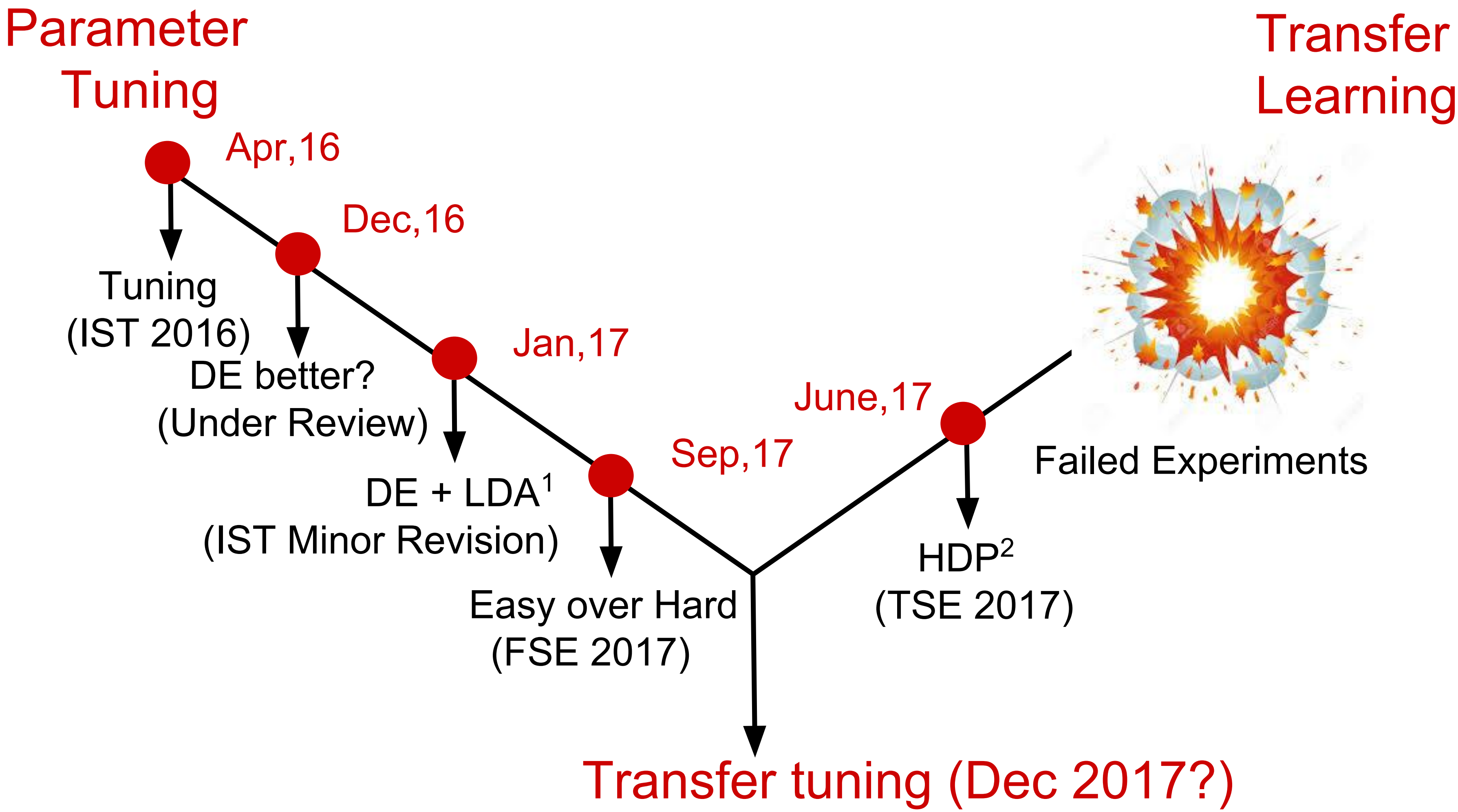
Transfer Learning

Failed Experiments 

- HDP[2](TSE 17) June,17

1. This is a joint work with Amrit Agrawal
2. This is a joint work with Dr. JC Nam from Waterloo University.

Plan of work



1. This is a joint work with Amrit Agrawal
2. This is a joint work with Dr. JC Nam from Waterloo University.

Q & A

- *Is tuning with DE helpful?*
  - Tuning for defect predictors (**IST'16**)
  - Tuning for topic modeling (**IST**, minor revision)
- *Is tuning with DE a faster method?*
  - DE v.s. grid search (under review)
  - DE+SVM v.s. deep learning (**FSE'17**)
- *How to improve tuning with DE?*
  - Future work...

Heterogeneous Defect Prediction
(**TSE**'17)

JIT Effort-aware Defect Prediction
(**FSE**'17)

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.
19. [Klein'2017 AISTATS]Klein, Aaron, et al. "Fast bayesian optimization of machine learning hyperparameters on large datasets." *arXiv preprint arXiv:1605.07079* (2016).
20. [Li'2017ICLR]Li, Lisha, et al. "Hyperband: Bandit-based configuration evaluation for hyperparameter optimization." (2017) ICLR]