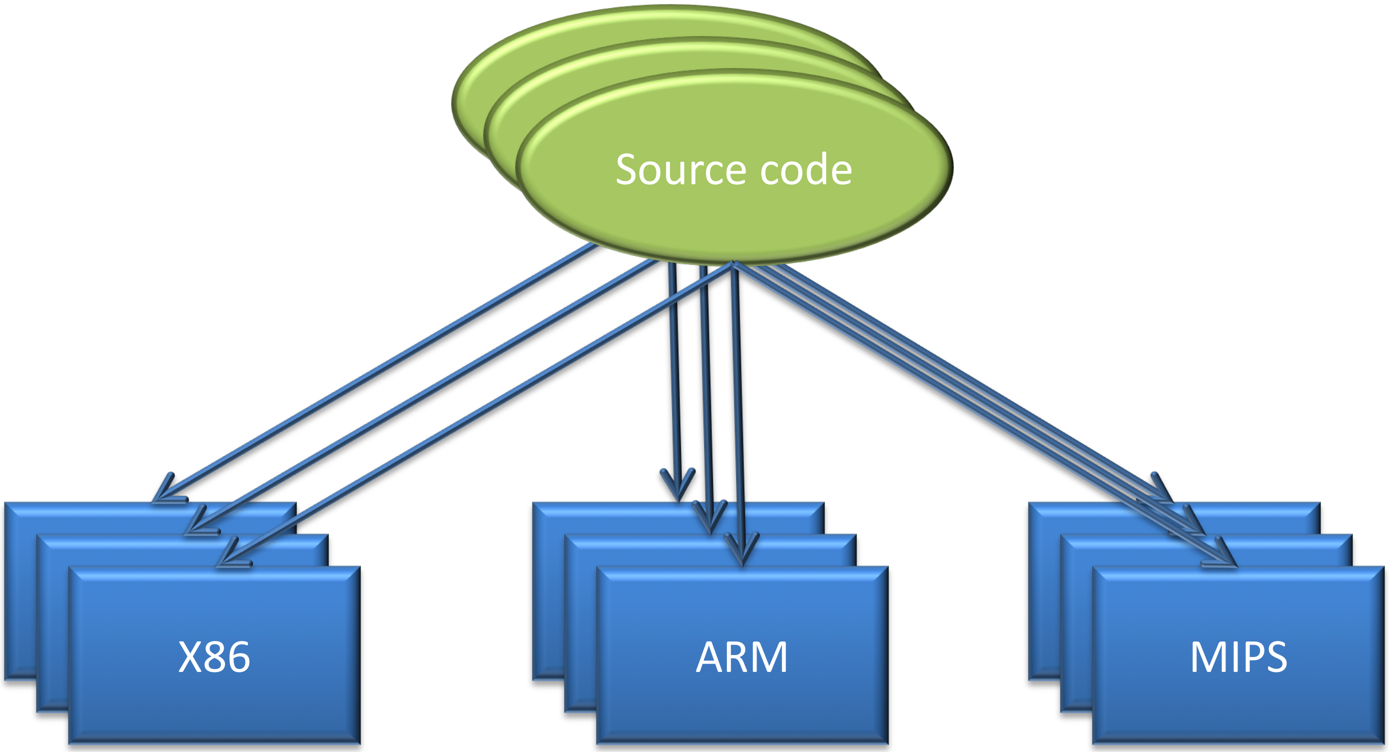
Neural Machine Translation Inspired Binary Code Similarity Comparison *beyond Function Pairs*

Fei Zuo, Xiaopeng Li, Patrick Young, Lannan Luo*,
Qiang Zeng*, Zhexin Zhang

NDSS 2019

UNIVERSITY OF
SOUTH CAROLINA

Why Cross-Architecture Binary Code Similarity Comparison?



Source code
X86
ARM
MIPS

Cross-architecture binary code similarity comparison

- **Plagiarism detection**
- **Malware family identification**
- **Vulnerability discovery**

A challenging task due to **different**

- **Instruction sets**
- **Registers**
- **Memory addressing**
- **Calling conventions**
- **Compilation optimizations**
- **…**

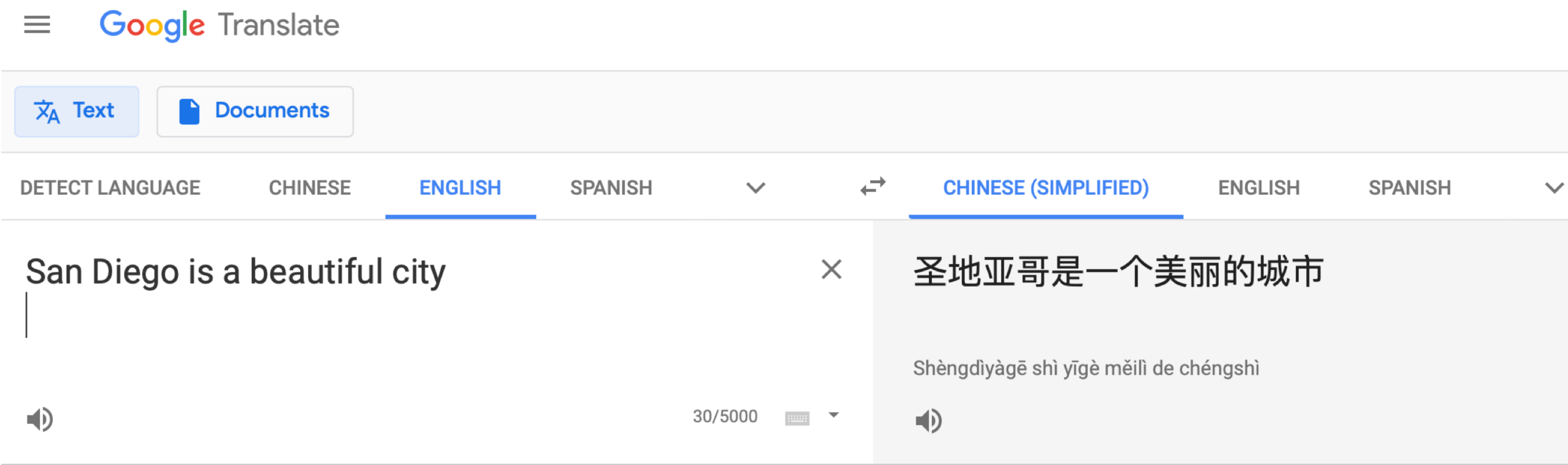
What is the current research status?

- Non-machine-learning approaches
  - Multi-MH [S&P'15]: fuzzing (on basic blocks)
  - Esh [PLDI'16]: SMT (on IR)
  - David et al. [PLDI'17]: re-compilation (of IR)
  - ➢ *Slow*

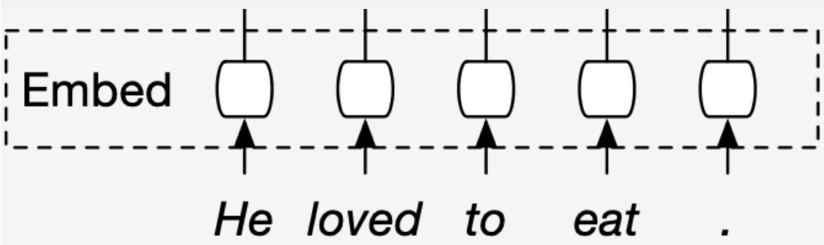
- Machine-learning approaches
  - Genius [CCS'16]: traditional ML
  - Gemini [CCS'17]: deep learning (graph)
  - ➢ *Fast; accurate (at function level)*
  - ➢ *But…*

Gemini used some manually selected features to represent a basic block, e.g., # of instructions, # calls, etc.

- Is it good enough?
  - Basic-block comparison: AUC = 0.85
- Could we do better?
- What information is lost?
  - Instruction meaning
  - Instruction dependence



- *Neural Machine Translation*: deep learning for translation
- First proposed in 2014
- Already adopted by Google and Microsoft



A binary, after disassembly, is represented in some assembly language. Can NMT handle assembly languages as well?

More specifically, given that NMT can translate sentences, can it also compare code of different architectures?

Image credit: smerity.com

Interesting idea, but tons of questions

- Words ⇔ instructions, but an infinite vocabulary?
  - E.g., mov edx, 200
- Sentences ⇔ basic blocks vs. functions?
  - A sentence: a sequence of words
  - A basic block: a sequence of instructions
  - A function: a graph
- Corpus of equivalent basic block pairs?
  - Unlike functions, which have names
- Expensive hardware?
  - We are not Google
  - Would be impractical if expensive facilities are required
- Interesting application?
  - Submitted to S&P in 05/2018; comment: no interesting application

```
MOVL %ESI, $.L.STR.31
MOVL %EDX, $3
MOVQ %RDI, %RAX
CALLQ STRNCMP
TESTL %EAX, %EAX
JE .LBB0_5
```

Instruction preprocessing:

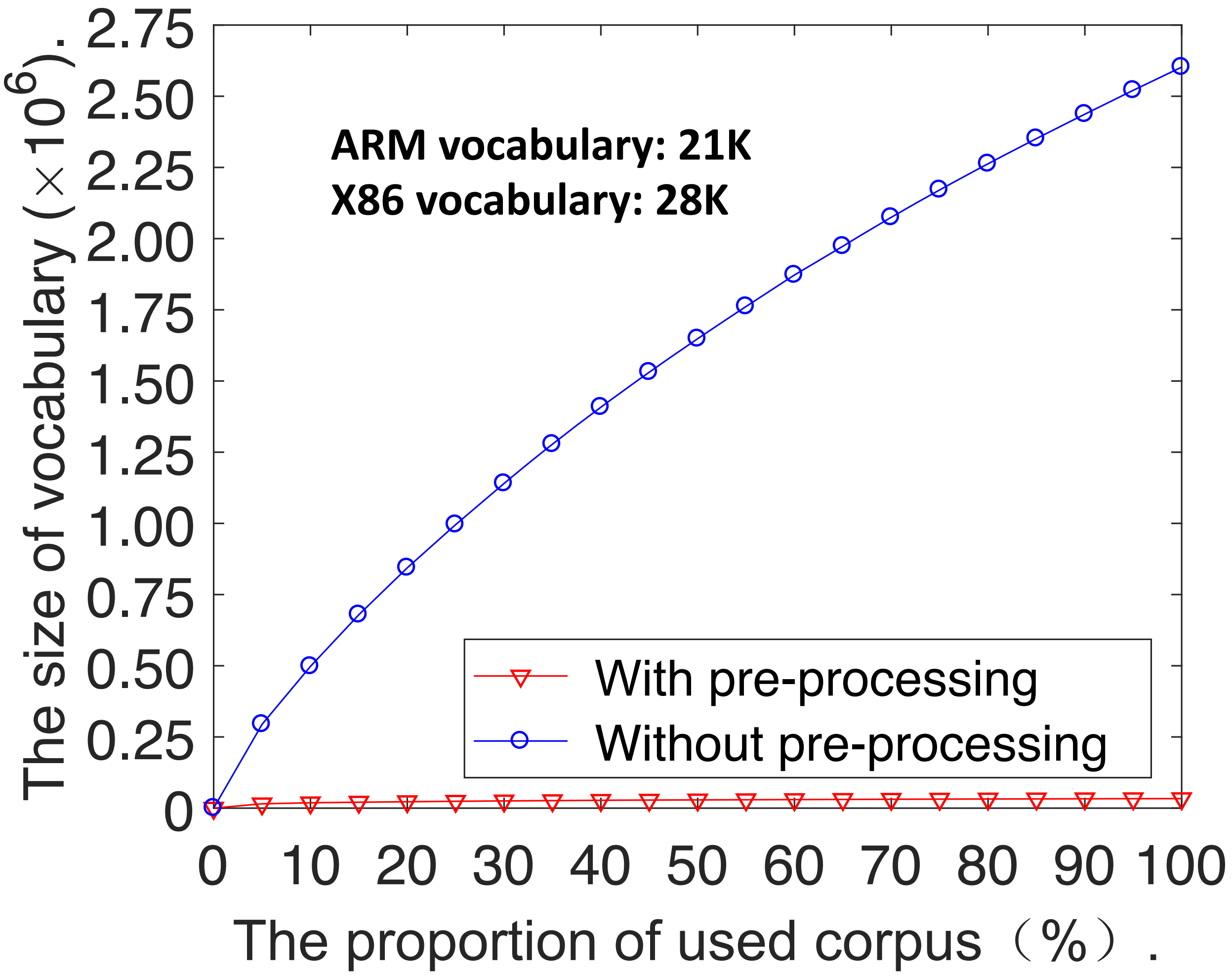
(1) Constant value => 0

(2) Strings => <str>

(3) Function names => FOO

(4) Other labes => <TAG>

Vocabulary size



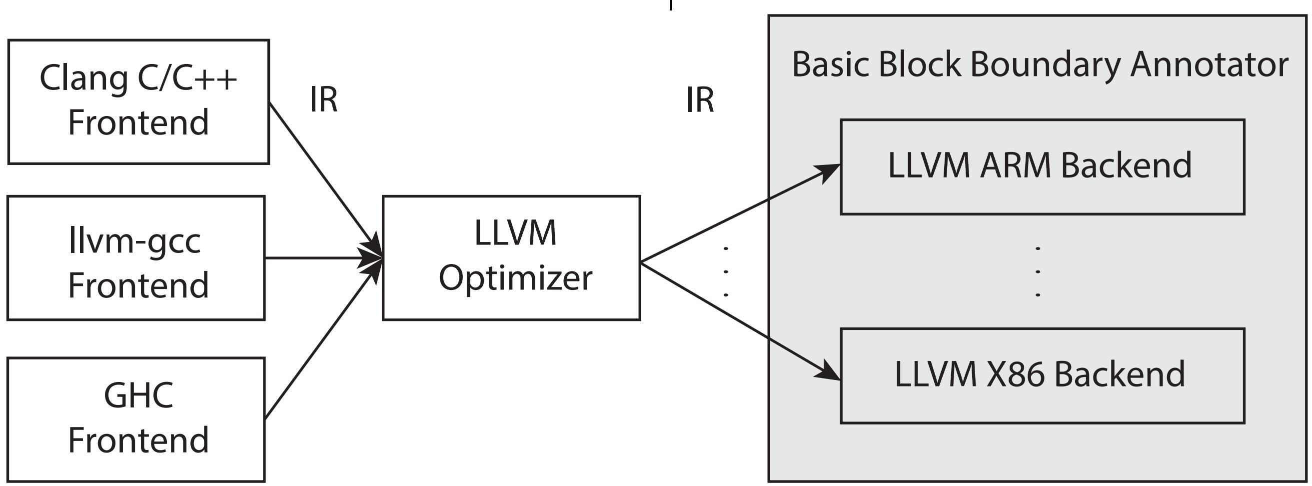
The *word2vec* network is then trained using the preprocessed instructions

Then, the network is used to convert each instruction into an *instruction embedding*

Corpus of equivalent BB pairs

```
MOVSLQ RSI,EBP
MOVZBL ECX,[R14,RBX]
MOVL EDX,<STR>
XORL EAX,EAX
MOVQ RDI,R13
CALLQ FOO
TESTL EAX,EAX
JLE <TAG>
```

```
LDRB R0,[R8+R4]
STR R9,[SP]
STR R0,[SP+0]
ASR R3,R7,0
MOV R0,R6
MOV R2,R7
BL FOO
CMP R0,0
BLT <TAG>
```



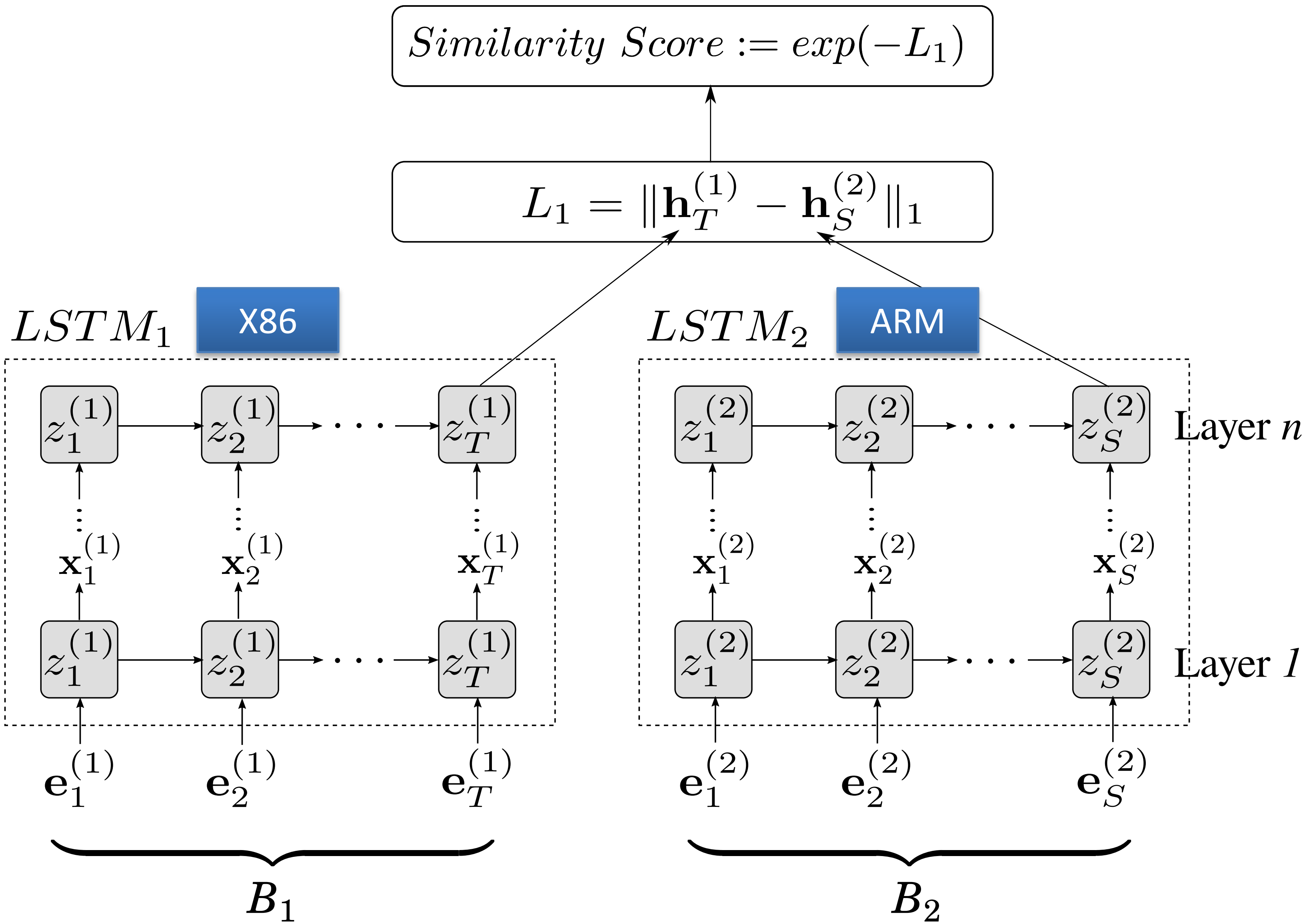
At backends, BBs generated from the same ***IR BB*** obtain the same annotated ID

Architecture for cross-architecture BB similarity comparison

X86 => ARM, then compare two ARM BBs?

- No, NLP researchers use the *Siamese architecture* to compare the similarity of two sentences [AAAI'16]
  - Mueller et al. "Siamese recurrent architectures for learning sentence similarity." AAAI 2016.

Architecture for cross-architecture BB similarity comparison



Interesting Application

- Prior cross-architecture binary analysis
  - answers whether *C1* is *equivalent* to *C2*
  - cannot answer whether *C1* is *contained* in program *P*
- The code containment problem:
  - Vulnerable code is inlined as part of another function
  - An attacker reuses a crypto in multiple malware
  - One steals a piece of code and inserts it into program
  - …
- *Not explored yet* in cross-architecture scenarios

- To determine whether *C* is contained in *P*
  - The CFG of *C* is decomposed into multiple paths
  - For each path *x* of *C*, LCS (longest common subsequence) and breadth-first search are combined to search in the CFG of *P*, and calculate a score for path *x*
  - Based on all path scores, a final score is calculated

- It was proposed by Luo et al. [FSE'14]
  - Symbolic execution for BB comparison
  - Mono-architecture code analysis

- Applying our NMT-based BB comparison
  - The first solution to cross-architecture code containment
  - Much faster

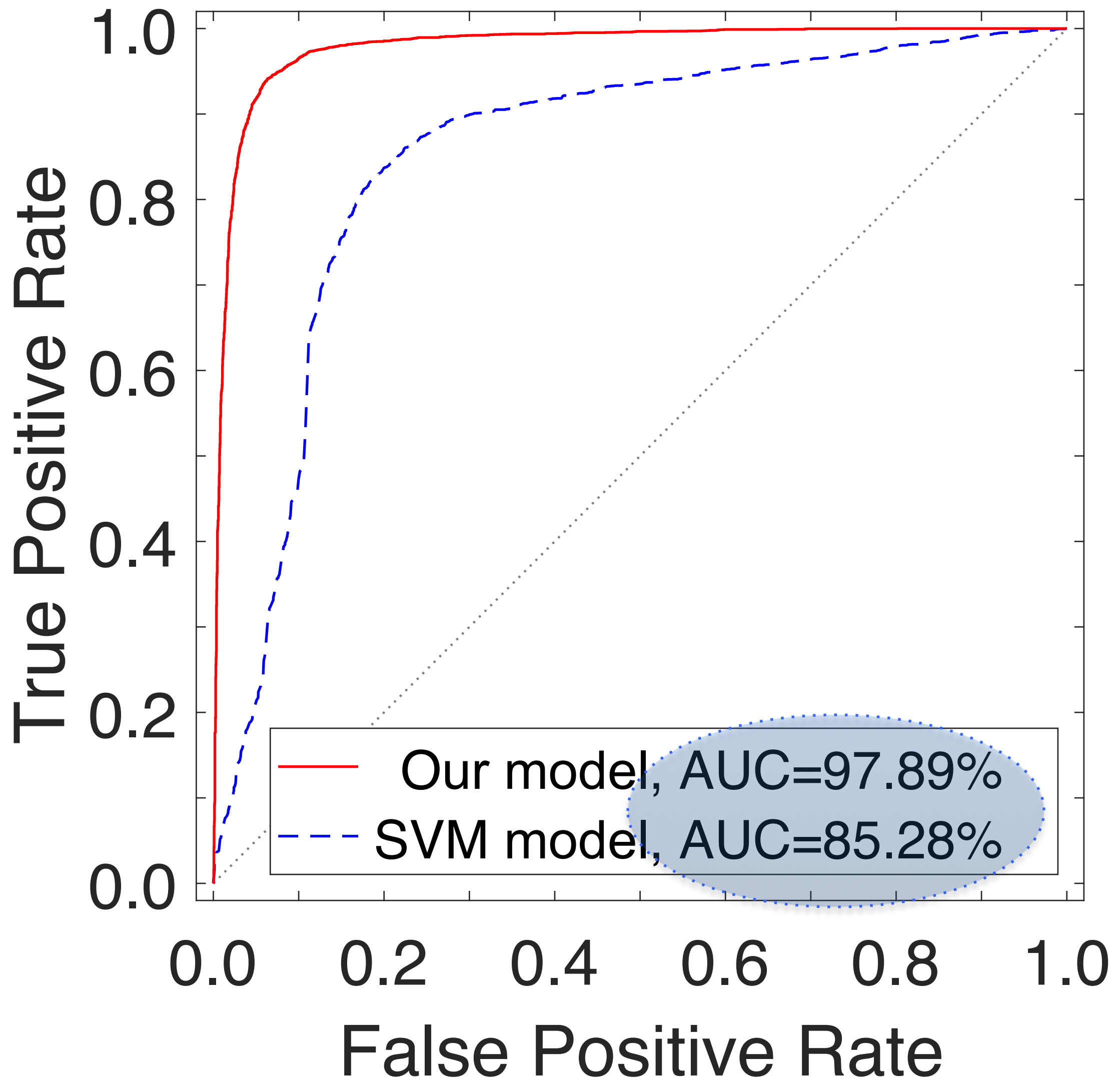
Hardware

- Actually, a Dell laptop
  - 2.7 GHz Intel i7
  - 32 GB RAM
  - No GPUs

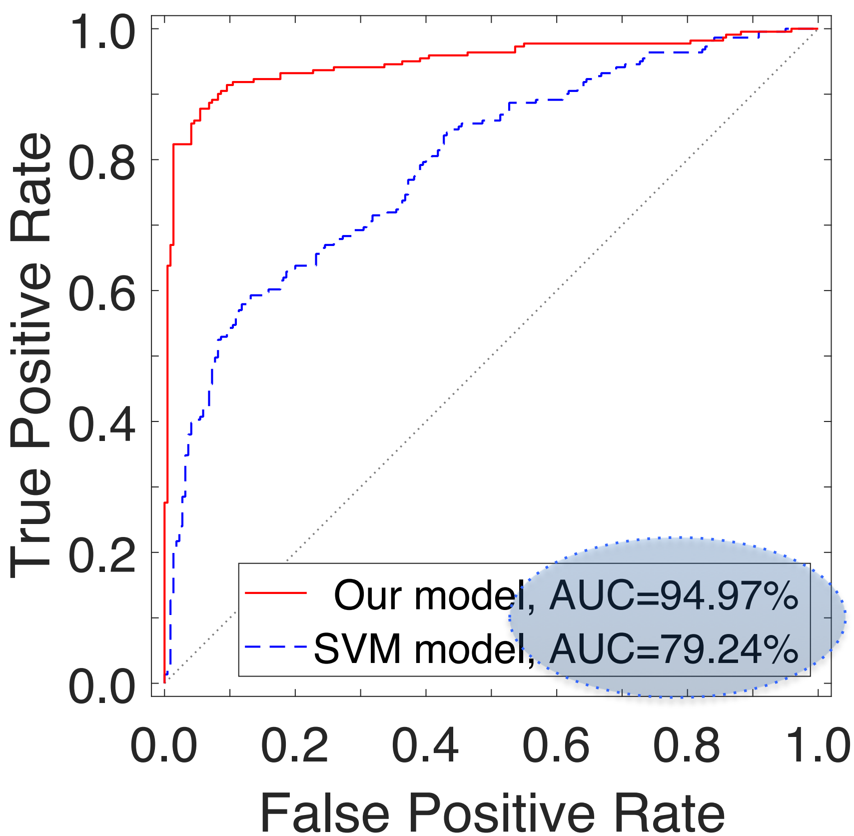
Datasets for training InnerEye-BB

	Total		
	Sim.	**Dissim.**	**Total**
O1	43,686	43,523	87,209
O2	56,082	55,937	112,019
O3	60,003	59,857	119,860
Cross-opts	42,481	42,074	84,555
Total	202,252	201,391	403,643

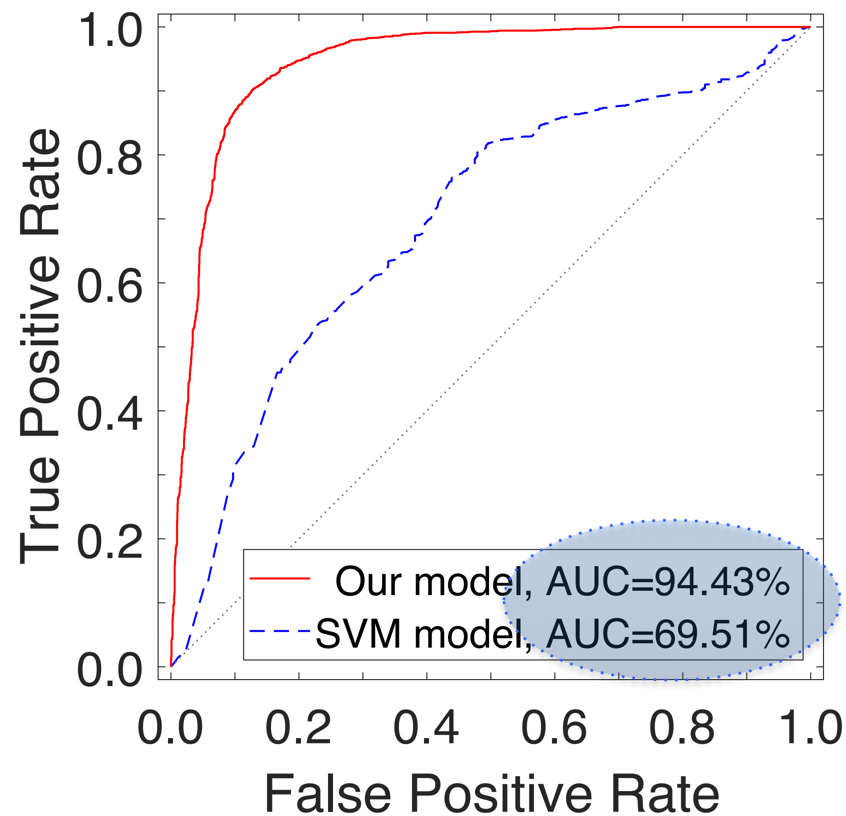
- Training : validation : testing = 0.8 : 0.1 : 0.1
- **Deduplication**: any BB in training does not re-appear in validation or testing



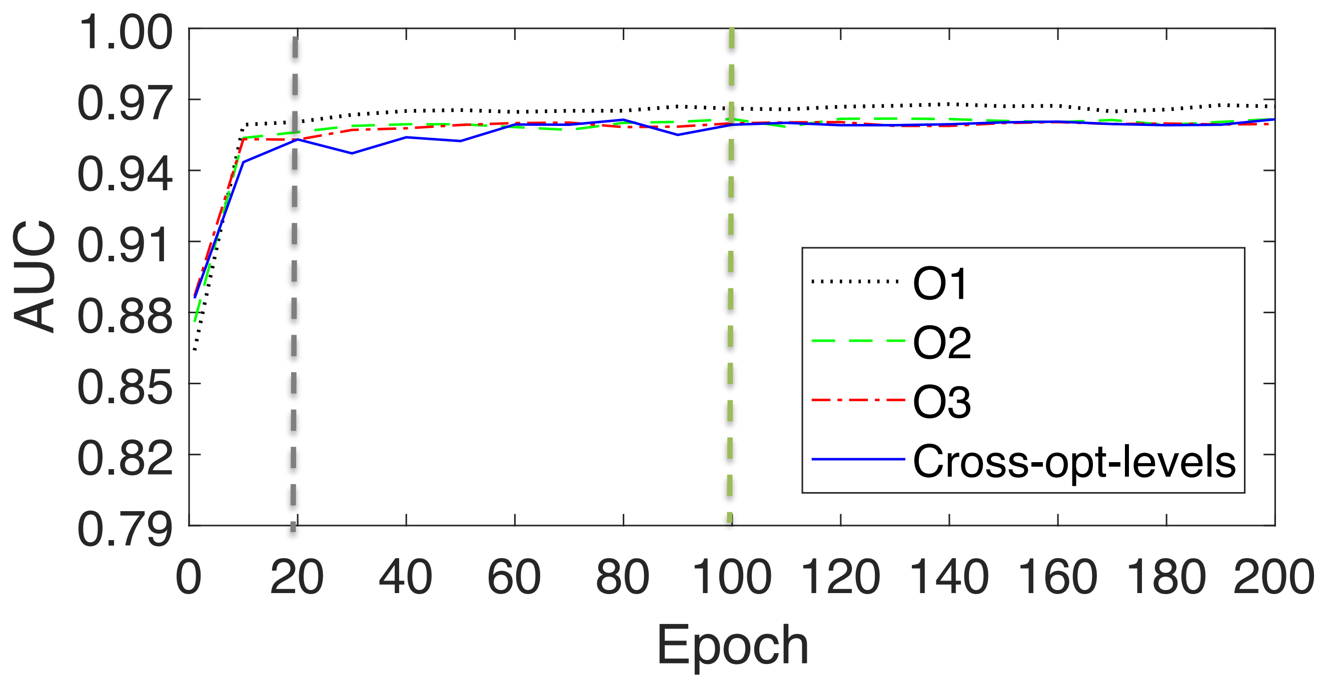
Cross-optimization levels, different sizes of BBs



Large BBs at O3



Small BBs at O3



Good accuracy after 20 epochs

Each epoch takes 971 seconds

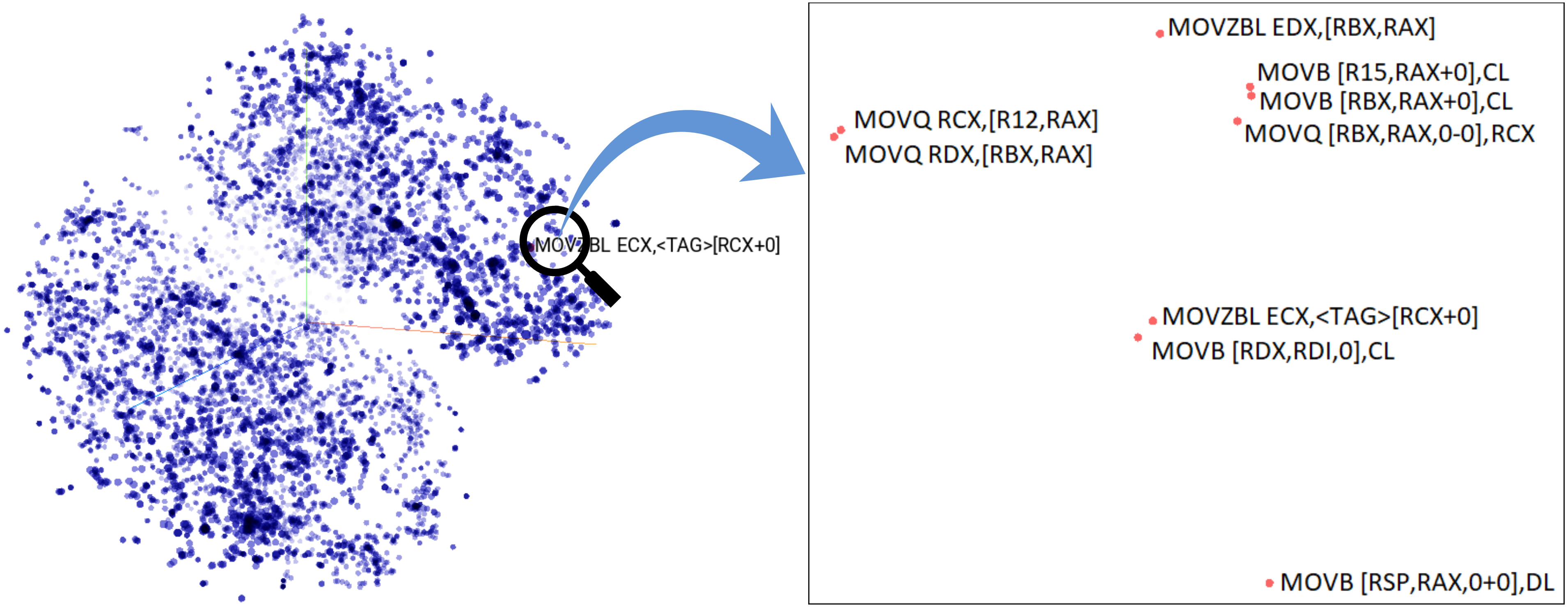
Training time: 5.5 hours

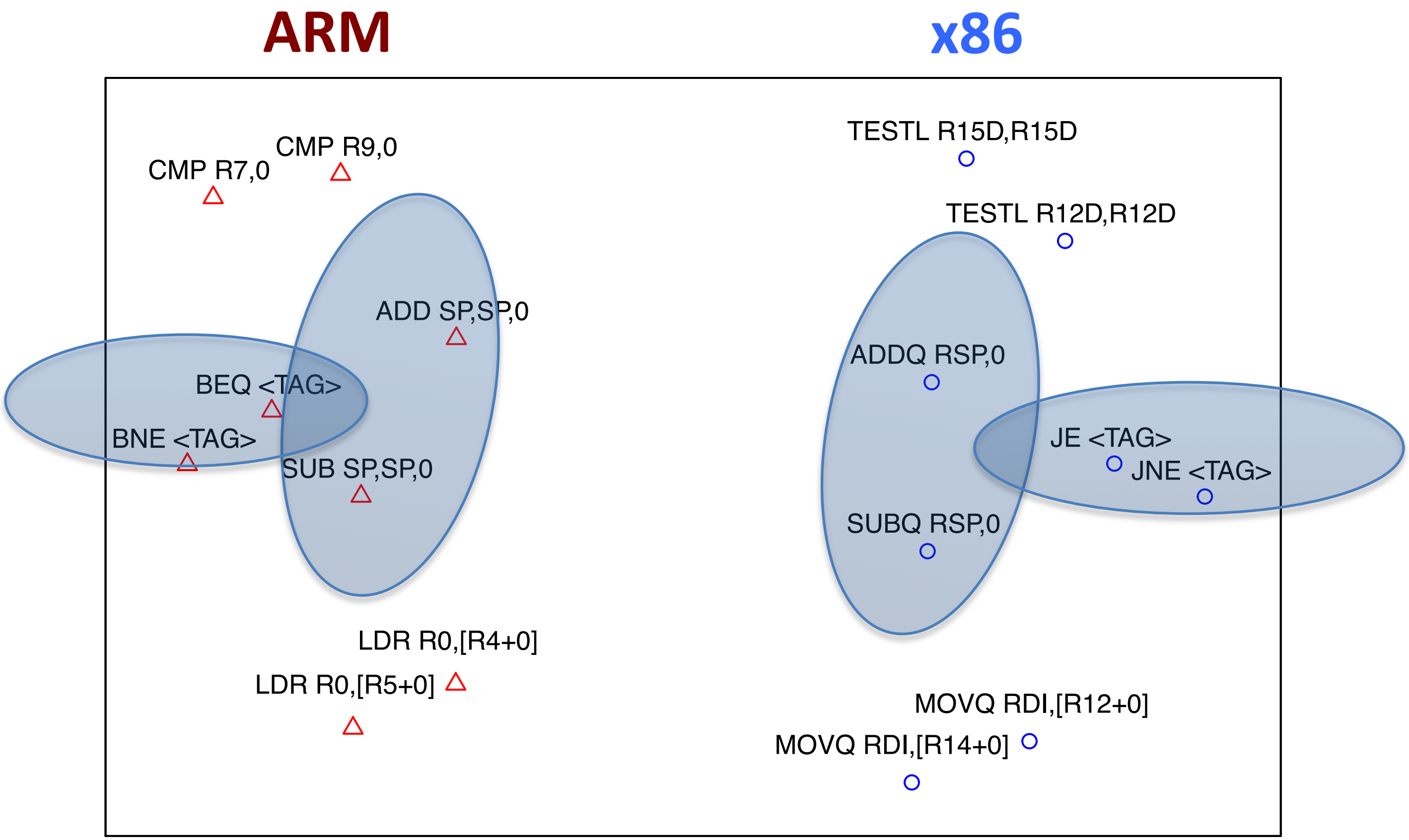
Testing time per BB pair: 0.76 ms

Case studies on code containment

- Whether the *URL checking loop* of *thttpd* is contained in other programs
  - *sthttpd* got a score 0.91, while others got < 0.04
  - Consistent with manual checking
- Whether MD5 code of *OpenSSL* is included in other 12 programs
  - High scores (0.88~0.93) for *cryptlib, openssh, libgcrypt,* etc.
  - Low scores for others

t-SNE of instructions





- A **good** word embedding model
  - cos (**“man”, “woman”**) ≈ cos (**“king”, “queen”**)

- Our instruction embedding model
  - cos(**BEQ <TAG>, BNE <TAG>**) ≈ cos(**JE <TAG>, JNE <TAG>**)
  - cos(**{ADD SP,SP,0}, {SUB SP,SP,0}**) ≈ cos(**{ADDQ RSP,0}, {SUBQ RSP,0}**)

Take-away messages

- NMT-inspired cross-architecture binary code similarity comparison works well (AUC = 0.98)
  - *Can NLP inspire us (binary analysts) more?*
- Does not need "big data" (400k samples)
- A laptop without GPU can do the job
- First solution to cross-arch code containment

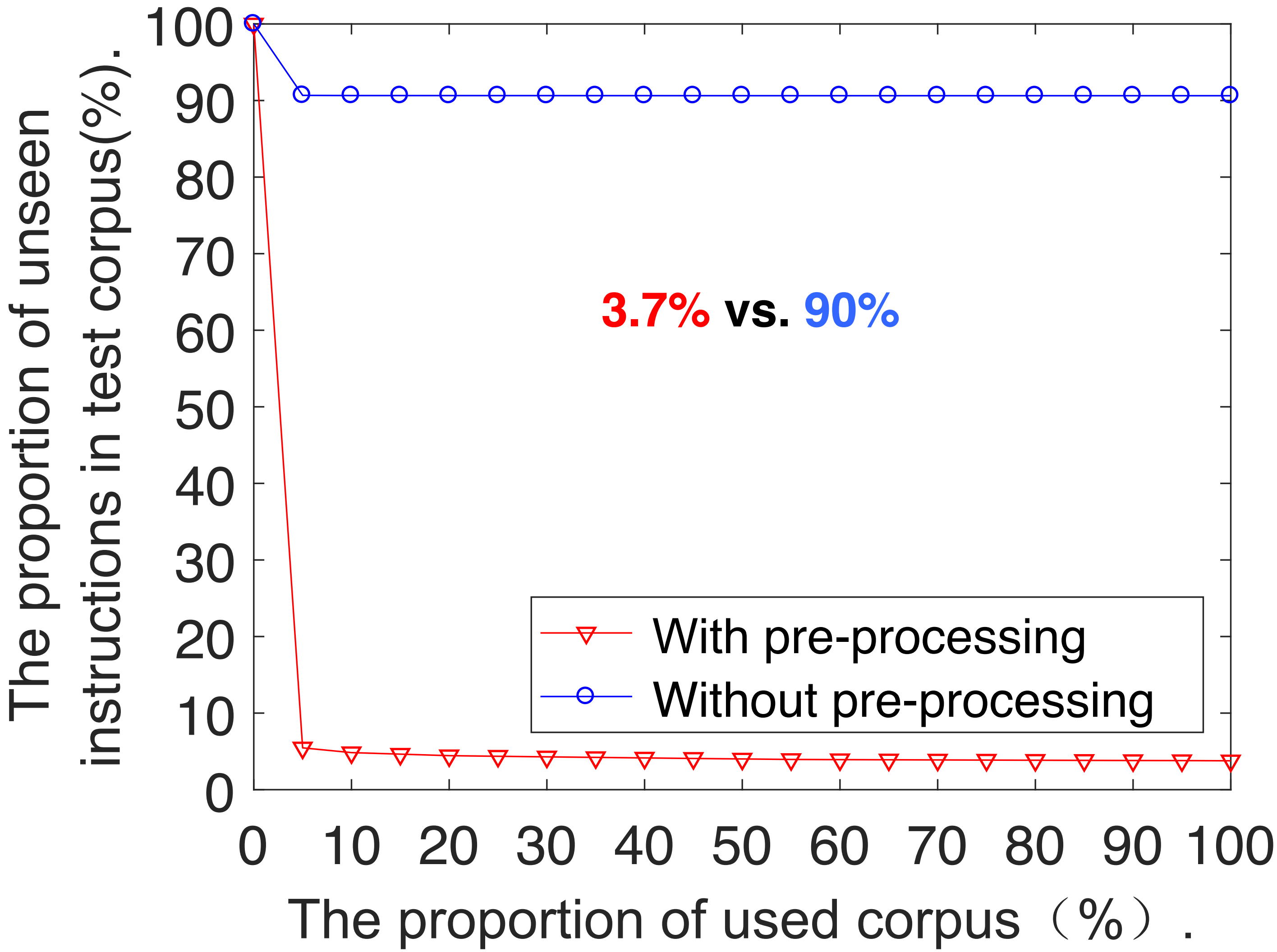
- Uncertain: cross-compiler? (on-going work) https://nmt4binaries.github.io (online since August 2018)

Contact: Qiang Zeng (zeng1@cse.sc.edu)

Thank you!
Q&A

UNIVERSITY OF SOUTH CAROLINA

Out-of-Vocabulary (OOV) rate



How about BBs of different optimization levels of the same architecture?

- O3 B1 => O0 B2 => src code
- Compare src code of B1 and B2
- If they are the same, B1 and B2 are *similar*

How about dissimilar BB pairs?

- ARM O3 BB1 => ARM O0 BB2
- X86 O2 BB3 => X86 O0 BB4 => ARM O0 BB5
- Use ***n-gram*** to compare BB2 and BB5
- If they are dissimilar, BB1 and BB3 are dissimilar

Interesting idea, but tons of questions

- Words ⇔ instructions, but an infinite vocabulary?
- Sentences ⇔ basic blocks vs. functions?
- Corpus of equivalent basic block pairs?
- Architecture?
- Expensive hardware?
- Interesting applications?

- Please refer to our paper for more details
  - Ground truth of dissimilar BB pairs
  - Selection of many hyperparameters
  - …