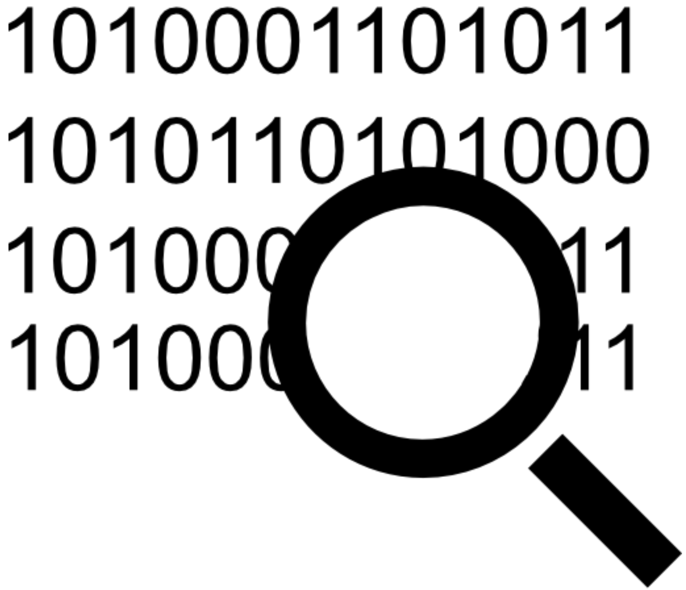
Binary analysis: Code Search

Yue Duan

Outline

- Code Search basics
- Research papers:
    - Scalable Graph-based Bug Search for Firmware Images
    - Neural Network-based Graph Embedding for Cross-Platform Binary Code Similarity Detection
    - DeepBinDiff: Learning Program-Wide Code Representations for Binary Diffing



Problem definition

- given two pieces of binary code (e.g., binary functions)
    - maybe in different architectures
    - maybe by different compilation configs
        - compilers
        - compiler versions
        - optimization levels
        - other options
  - check if they are semantically equivalent or similar

Security applications

- plagiarism detection

```
static void
make_blank (struct line *blank, int count)
{
  int i;
  unsigned char *buffer;
  struct field *fields;

  blank->nfields = count;
  blank->buf.size = blank->buf.length = count + 1;
  blank->buf.buffer = (char*) xmalloc (blank->buf.size);
  buffer = (unsigned char *) blank->buf.buffer;
  blank->fields = fields =
    (struct field *) xmalloc (sizeof (struct field) * count);

  for (i = 0; i < count; i++){
      ...
  }
}
```

Original Code

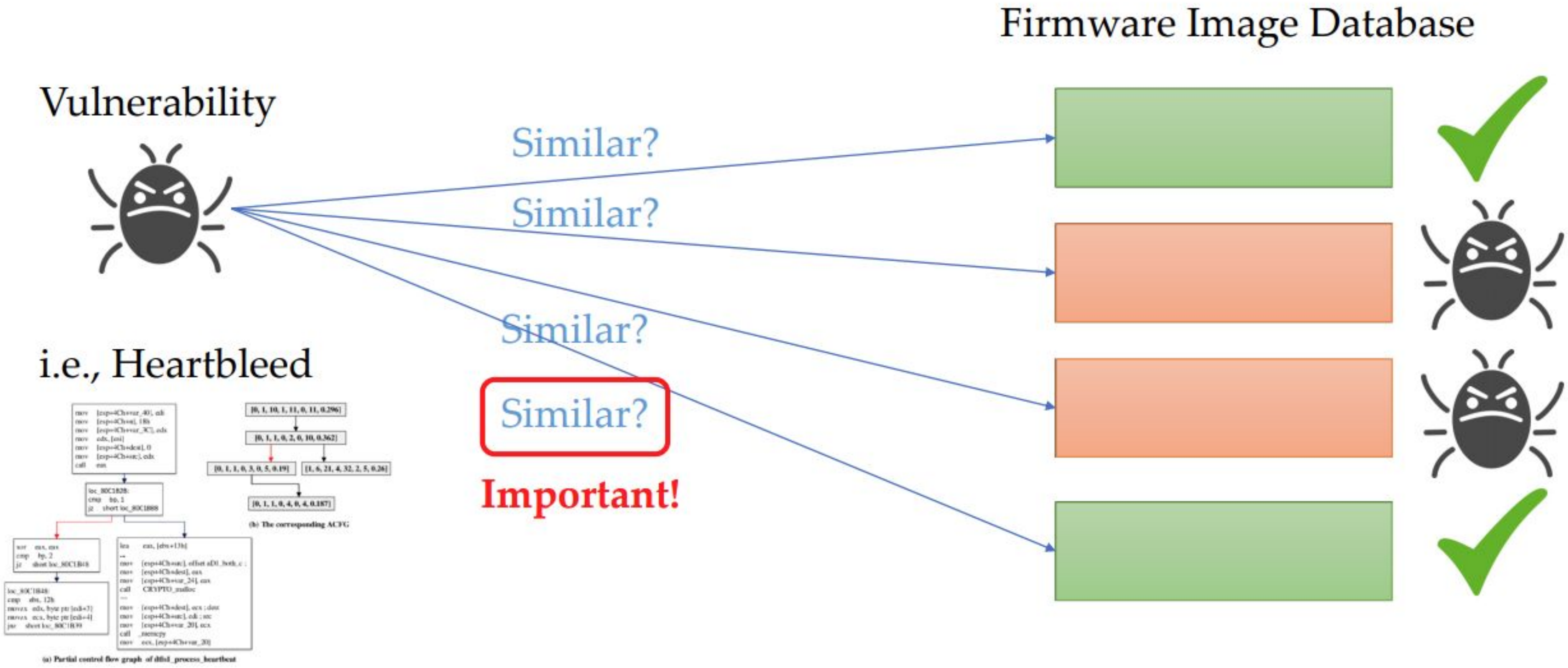
```
static void
fill_content(int num, struct line* fill)
{
  (*fill).store.size = fill->store.length = num + 1;
  struct field *tabs;
  (*fill).fields = tabs = (struct field *)
                xmalloc (sizeof (struct field) * num);
  (*fill).store.buffer = (char*) xmalloc (fill->store.size);
  (*fill).ntabs = num;
  unsigned char *pb;
  pb = (unsigned char *) (*fill).store.buffer;

  int idx = 0;
  while(idx < num){ // fill in the storage
    ...
    for(int j = 0; j < idx; j++)
      ...
    idx++;
  }
}
```

Plagiarized Code

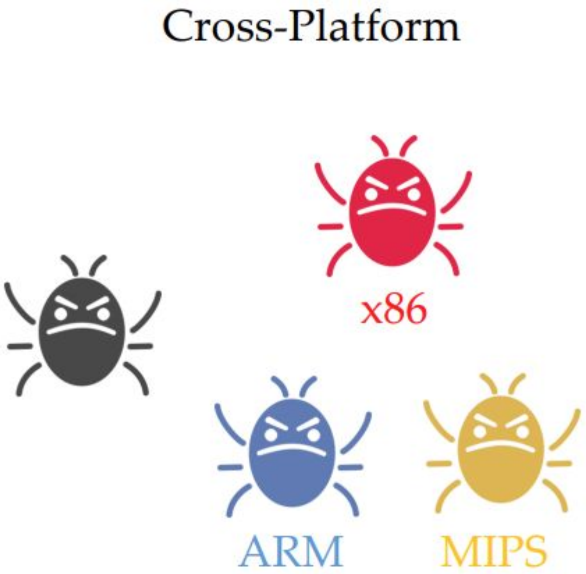
Security applications

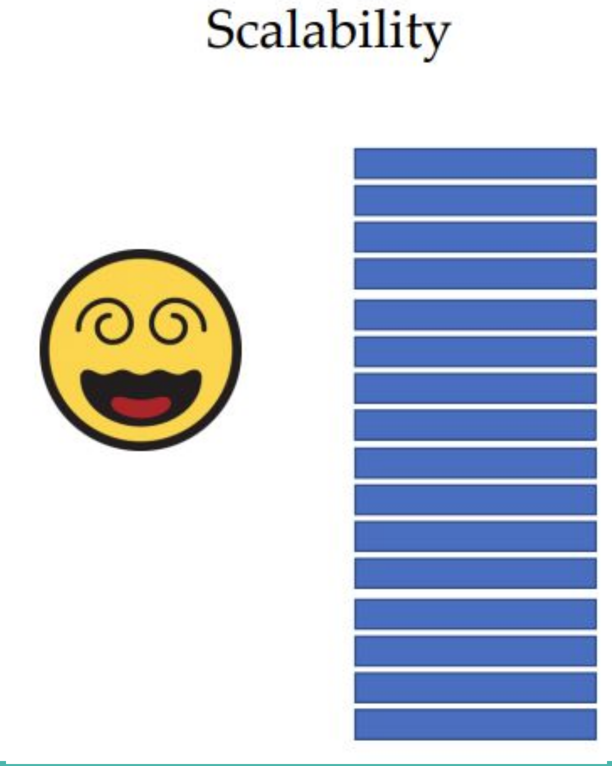
- vulnerability analysis
  - i.e., vulnerability search in IoT

Vulnerability analysis

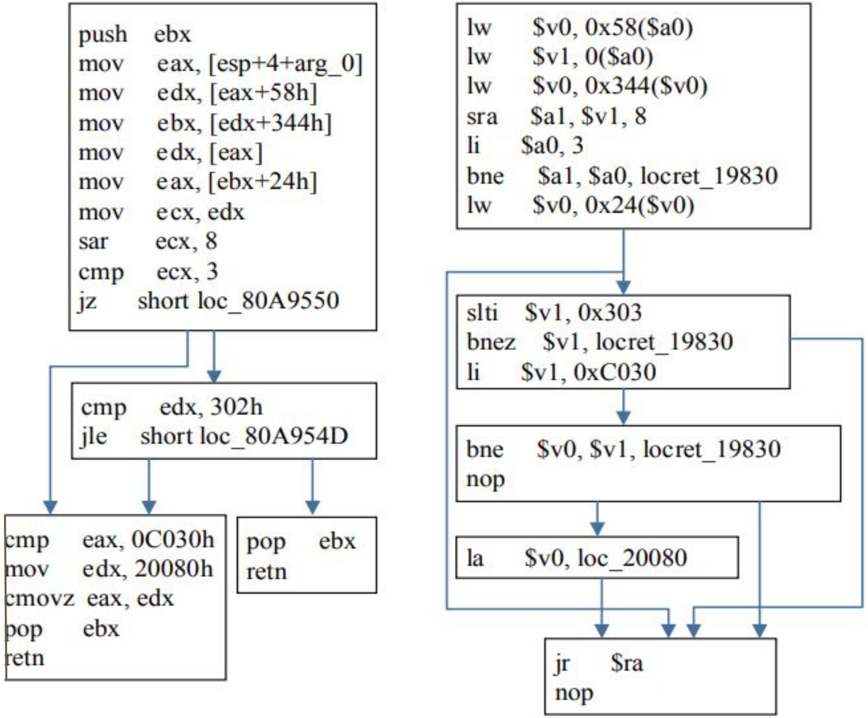


Challenges





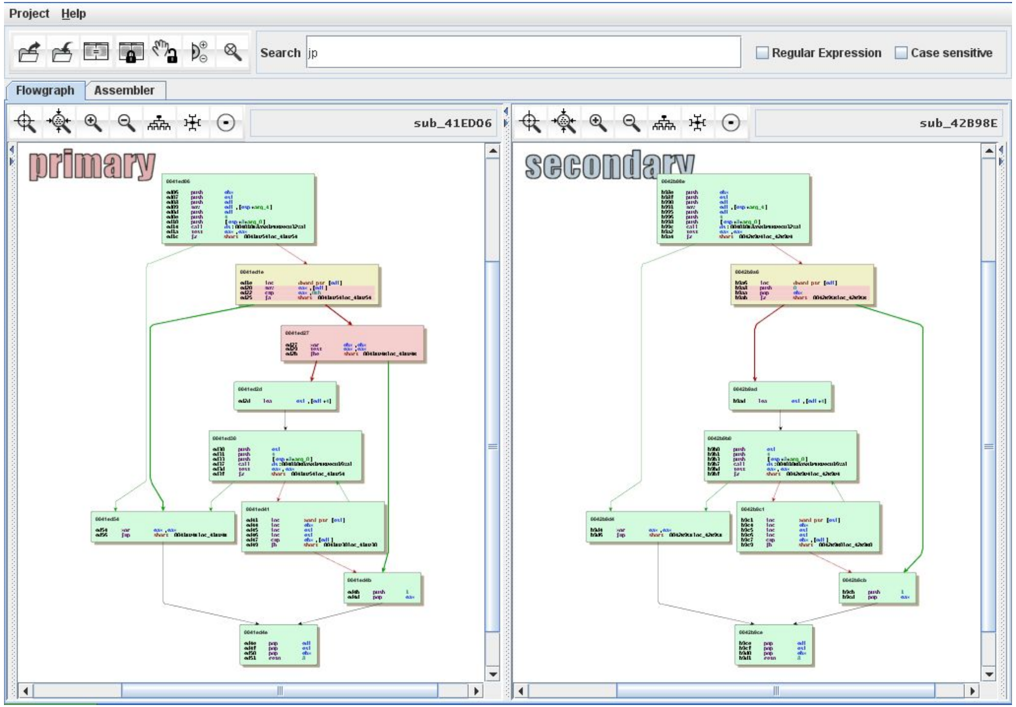
Example



a) x86 assembly **b) MIPS assembly**

Existing techniques - static approaches

- BinDiff https://www.zynamics.com/bindiff.html
  - de-facto commercial tool
  - binary => Control-flow graph
  - graph isomorphism detection
  - heuristics for runtime performance



Existing techniques - dynamic approaches

- Blanket execution [*USENIX Sec'14*]
  - dynamically execute two given binaries
  - collect runtime information during execution
  - checking the semantic level equivalence based on the information

Existing techniques

- static analysis
    - low accuracy
    - syntax rather than semantics
- dynamic analysis
    - code coverage issue
    - poor scalability

new trend - Learning-based approaches!

Scalable Graph-based Bug Search for Firmware Images

Qian Feng , Rundong Zhou , Chengcheng Xu , Yao Cheng , Brian Testa , and Heng Yin

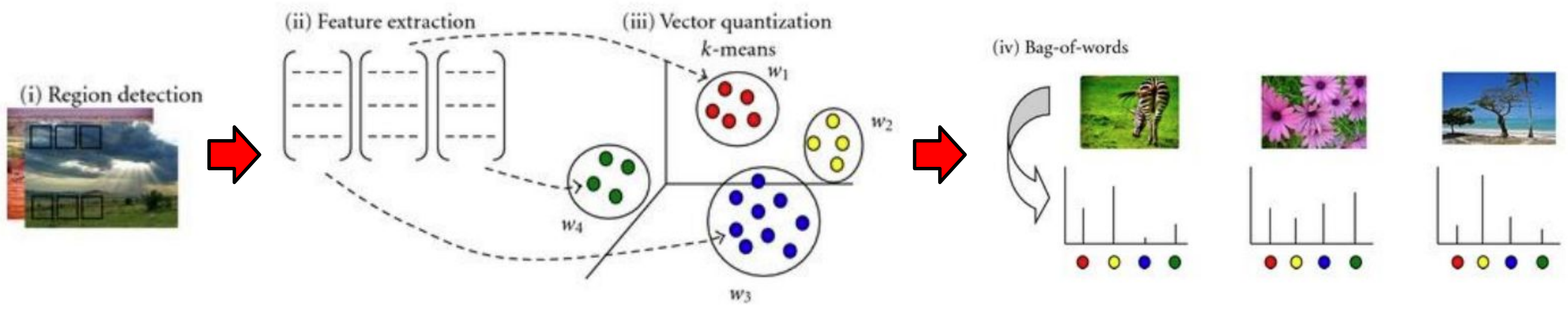
ACM CCS 2016

Motivation

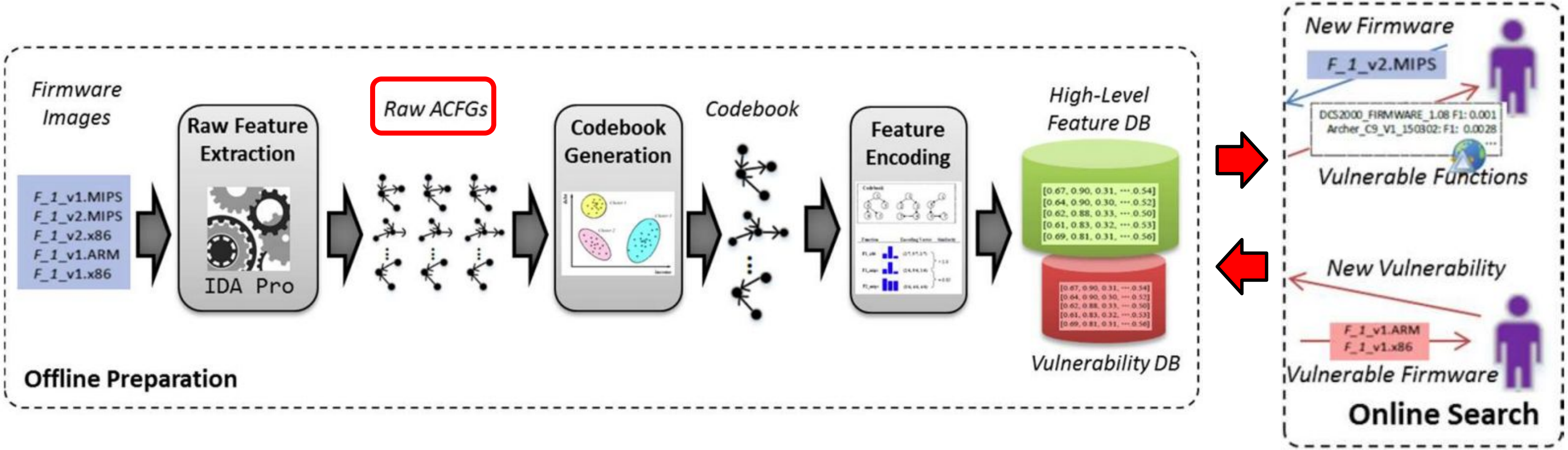
- Key challenge: cross-platform code search
  - string pattern or constant matching [Costin et al. USENIX Sec'14]
    - lack of generality
  - I/O pairs [Pewny et al. IEEE S&P'15]
    - lack of scalability
  - DiscovRe [Eschweiler et al. NDSS'16]
    - still not scalable enough
    - unreliable
- Biggest problem
  - graph matching is **EXPENSIVE!**
- **How to achieve high performance matching?**
  - **graph representation learning**

Graph Representation Learning

- Key idea:
  - learn from image processing
  - use a high-dimensional vector, a.k.a. embeddings, to represent a graph



Genius Overview



Attributed Control Flow Graph

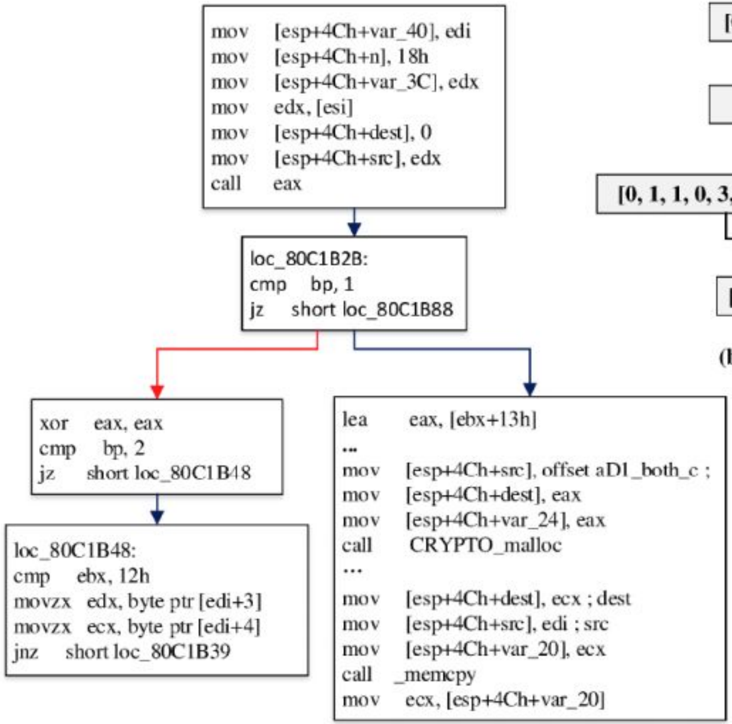
- Attributed Control Flow Graph
  - a control-flow graph with features

Table 1: Basic-block level features.

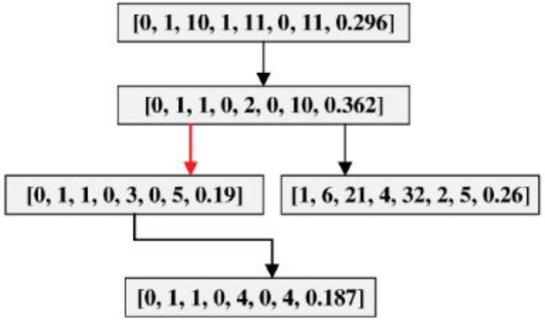
Type	Feature Name	Weight (α)
Statistical Features	String Constants	10.82
	Numeric Constants	14.47
	No. of Transfer Instructions	6.54
	No. of Calls	66.22
	No. of Instructions	41.37
	No. of Arithmetic Instructions	55.65
Structural Features	No. of offspring	198.67
	Betweeness	30.66

Attributed Control Flow Graph

- Example

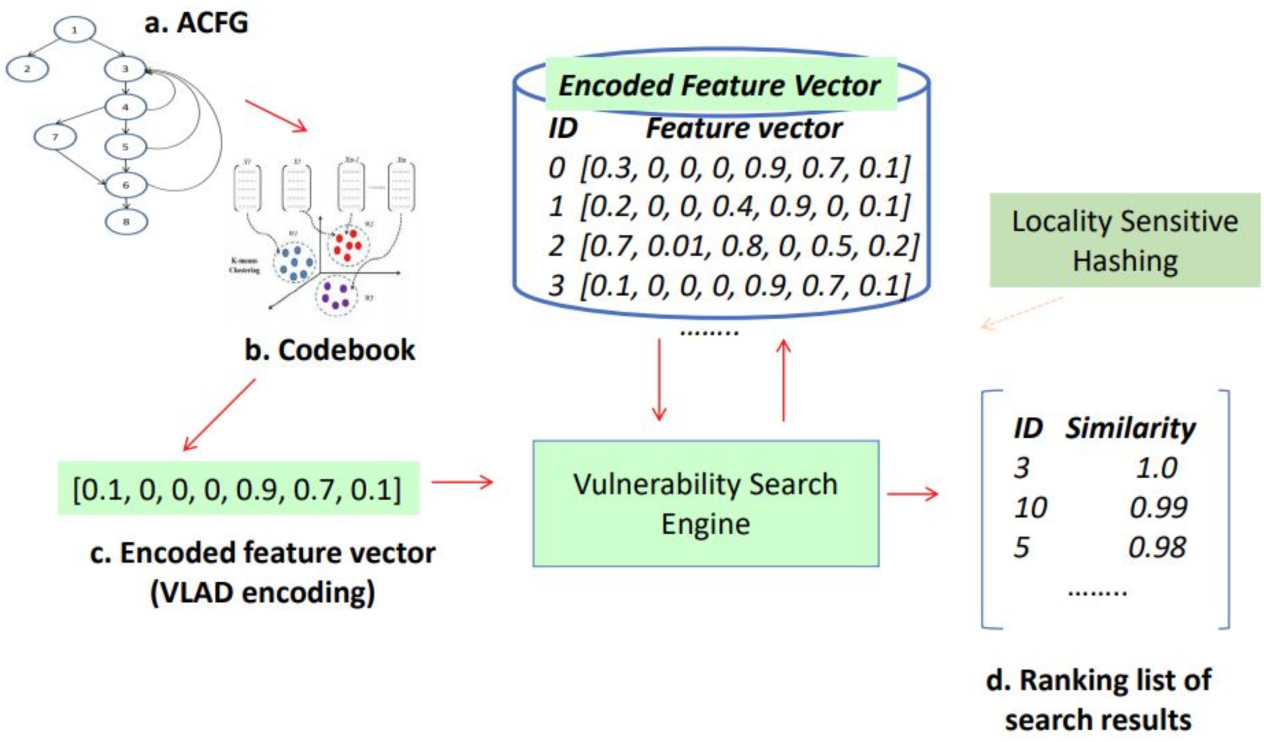


(a) Partial control flow graph of dtls1_process_heartbeat

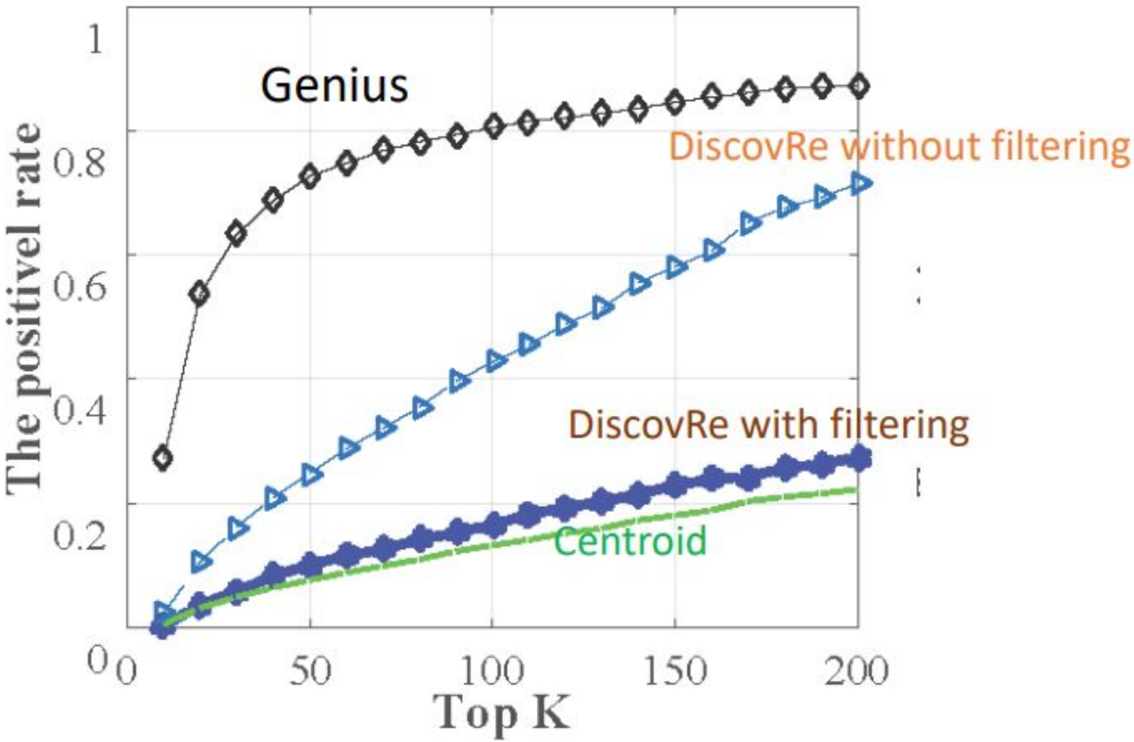


(b) The corresponding ACFG

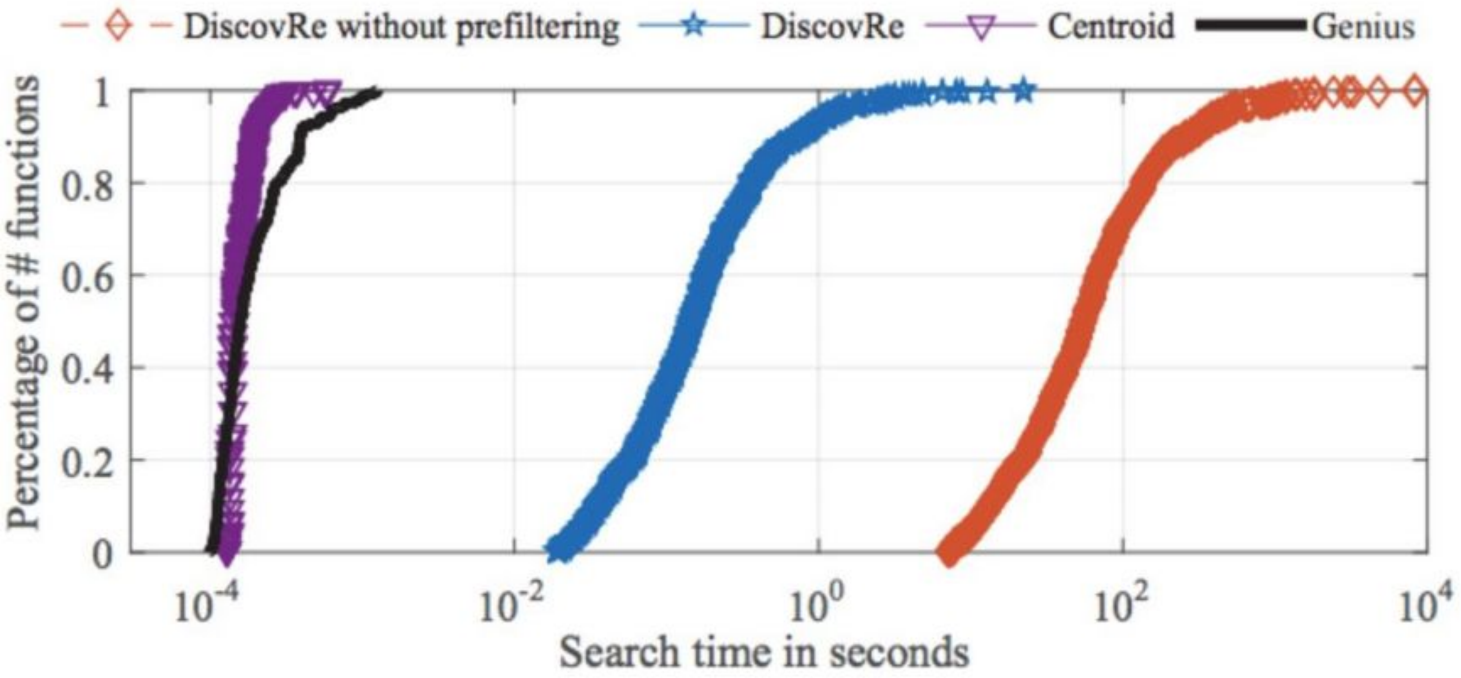
Index and Search



Evaluation: True positive rate



Evaluation: Efficiency



Neural Network-based Graph Embedding for Cross-Platform Binary Code Similarity Detection

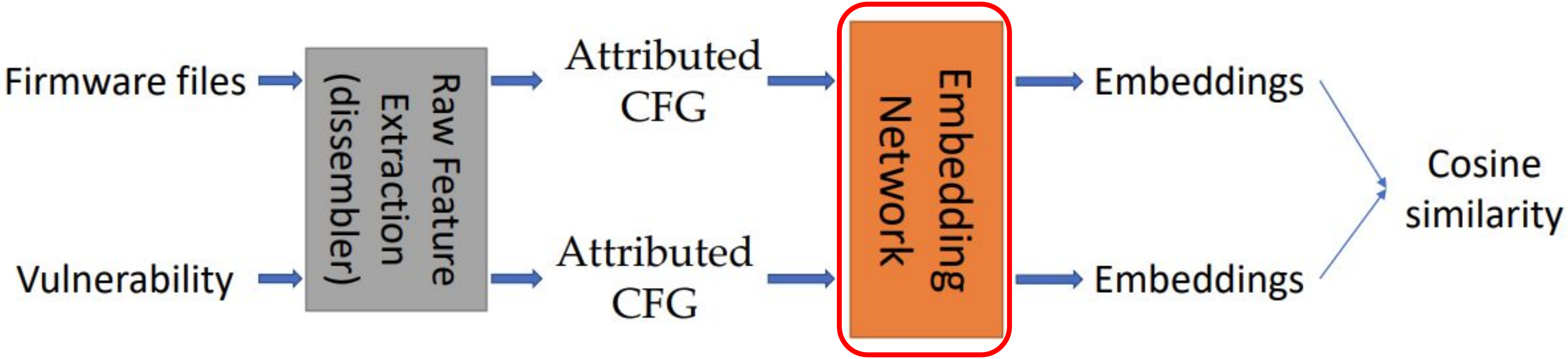
Xiaojun Xu, Chang Liu, Qian Feng, Heng Yin, Le Song, Dawn Song

ACM CCS 2017

Motivation

- Genius is great
  - more accurate: ACFG
  - faster: graph embeddings
- However
  - codebook generation can bring inaccuracy
  - graph representation learning is naive
- Key idea:
  - Can we use deep learning to learn graph embeddings?

Approach: Gemini



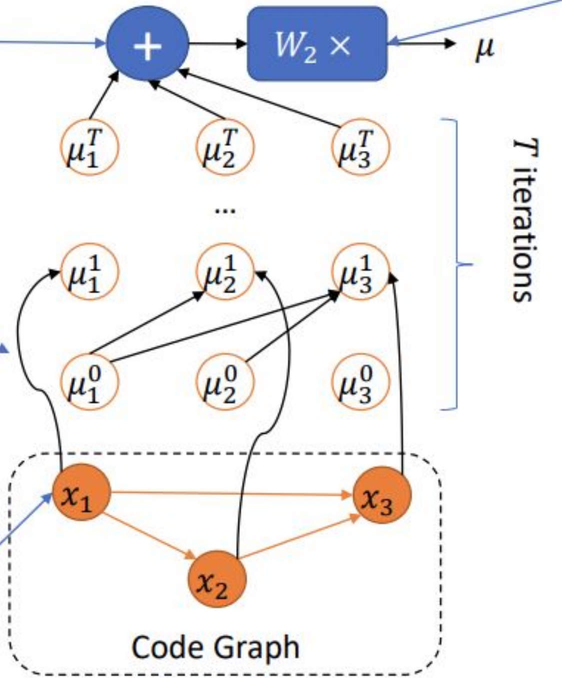
Embedding Network

3. After the last iteration, the embeddings on all vertexes are aggregated together

4. An affine transformation is applied in the end to compute the embedding for the graph

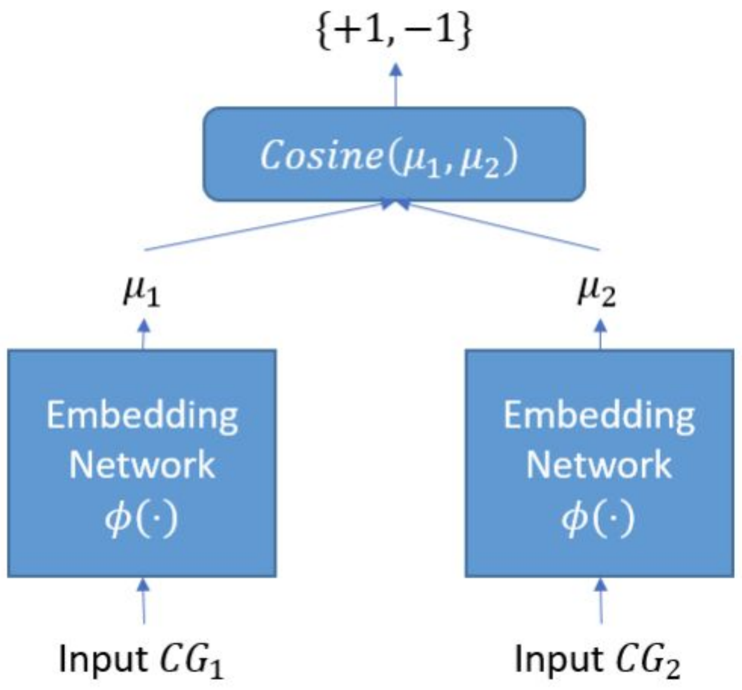
2. In each iteration, the embedding on each vertex is propagated to its neighbors

1. Initially, each vertex has an embedding vector computed from each code block



Training: Siamese Network

- Training data: a large number of functions
    - similar and dissimilar functions
- Train the network so that
    - similar functions will generate high similarity score
    - dissimilar functions will generate low similarity score



Evaluation: Efficiency

- Per function processing time
  - Genius: a few seconds to a few mins
  - Now: a few milliseconds

- Training time
  - Genius: > 1 week
  - Now: < 30 mins

Evaluation: Effectiveness

Function Name	Vendor	Firmware	Binary File	Similarity
ssl3_get_new_session_ticket	D-Link	DAP-1562_FIRMWARE_1.00	wpa_supplicant.acfgs	0.962374508
port_check_v6	D-Link	DES-1210-28_REVB_FIRMWARE_3.12.015	in.ftpd.acfgs	0.955408692
sub_42EE7C	TP-Link	TD-W8970B_V1_140624	racoon.acfgs	0.954742193
sub_42EE7C	TP-Link	TD-W8970_V1_130828	racoon.acfgs	0.954742193
prsa_parse_file	TP-Link	Archer_D5_V1_140804	racoon.acfgs	0.949814439
sub_432B8C	TP-Link	TD-W8970B_V1_140624	racoon.acfgs	0.949583828
sub_432B8C	TP-Link	TD-W8970_V1_130828	racoon.acfgs	0.949583828
ssl3_get_new_session_ticket	DD-wrt	dd-wrt.v24-23838_NEWD-2_K3.x_mega-WNR3500v2_VC	openvpn.acfgs	0.94668287
ucSetUsbipServer	TP-Link	WDR4900_V2_130115	httpd.acfgs	0.946312308
ssl3_get_new_session_ticket	Netgear	tomato-Cisco-M10v2-NVRAM32K-1.28.RT-N5x-MIPSR2-110-PL-Mini	libssl.so.1.0.0.acfgs	0.945933044
ssl3_get_new_session_ticket	Tomato_by_Shibby	tomato-K26-1.28.RT-MIPSR1-109-Mini	libssl.so.1.0.0.acfgs	0.945933044
ssl3_get_new_session_ticket	Tomato_by_Shibby	tomato-K26USB-1.28.RT-N5x-MIPSR2-110-VPN	libssl.so.1.0.0.acfgs	0.945932984
ssl3_get_new_session_ticket	Tomato_by_Shibby	tomato-E4200USB-NVRAM60K-1.28.RT-MIPSR2-110-PL-BT	libssl.so.1.0.0.acfgs	0.945932984
ssl3_get_new_session_ticket	Tomato_by_Shibby	tomato-E3000USB-NVRAM60K-1.28.RT-MIPSR2-110-BT-VPN	libssl.so.1.0.0.acfgs	0.945932984
ssl3_get_new_session_ticket	Tomato_by_Shibby	tomato-K26USB-1.28.RT-MIPSR1-109-AIO	libssl.so.1.0.0.acfgs	0.945932984
ssl3_get_new_session_ticket	Tomato_by_Shibby	tomato-Netgear-3500Lv2-K26USB-1.28.RT-N5x--109-AIO	libssl.so.1.0.0.acfgs	0.945932984
ssl3_get_new_session_ticket	Tomato_by_Shibby	tomato-E4200USB-NVRAM60K-1.28.RT-MIPSR2-109-AIO	libssl.so.1.0.0.acfgs	0.945932984
ssl3_get_new_session_ticket	Tomato_by_Shibby	tomato-E1550USB-NVRAM60K-1.28.RT-N5x-MIPSR2-110-Nocat-VPN	libssl.so.1.0.0.acfgs	0.945932984
ssl3_get_new_session_ticket	Tomato_by_Shibby	tomato-K26USB-1.28.RT-N5x-MIPSR2-115-PL-L600N	libssl.so.1.0.0.acfgs	0.945932984
ssl3_get_new_session_ticket	Tomato_by_Shibby	tomato-E1550USB-NVRAM60K-1.28.RT-N5x-MIPSR2-110-BT-VPN	libssl.so.1.0.0.acfgs	0.945932984
ssl3_get_new_session_ticket	Tomato_by_Shibby	tomato-E3000USB-NVRAM60K-1.28.RT-MIPSR2-108-PL-VPN	libssl.so.1.0.0.acfgs	0.945932984
ssl3_get_new_session_ticket	Tomato_by_Shibby	tomato-E1550USB-NVRAM60K-1.28.RT-N5x-MIPSR2-110-Mega-VPN	libssl.so.1.0.0.acfgs	0.945932984
ssl3_get_new_session_ticket	Tomato_by_Shibby	tomato-E1200v2-NVRAM64K-1.28.RT-N5x-MIPSR2-108-PL-Max	libssl.so.1.0.0.acfgs	0.945932984
ssl3_get_new_session_ticket	Tomato_by_Shibby	tomato-K26USB-1.28.RT-MIPSR1-109-Mega-VPN	libssl.so.1.0.0.acfgs	0.945932984
ssl3_get_new_session_ticket	Tomato_by_Shibby	tomato-E3000USB-NVRAM60K-1.28.RT-MIPSR2-109-Big-VPN	libssl.so.1.0.0.acfgs	0.945932984
ssl3_get_new_session_ticket	Tomato_by_Shibby	tomato-E4200USB-NVRAM60K-1.28.RT-MIPSR2-108-PL-Nocat-VPN	libssl.so.1.0.0.acfgs	0.945932984
ssl3_get_new_session_ticket	Tomato_by_Shibby	tomato-Netgear-3500Lv2-K26USB-1.28.RT-N5x--110-ND-AIO	libssl.so.1.0.0.acfgs	0.945932984
ssl3_get_new_session_ticket	Tomato_by_Shibby	tomato-E4200USB-NVRAM60K-1.28.RT-MIPSR2-109-Nocat-VPN	libssl.so.1.0.0.acfgs	0.945932984

- Among top 50 vulnerabilities
  - 42/50 can be identified
  - Genius: 10/50

DeepBinDiff: Learning Program-Wide Code Representations for Binary Diffing

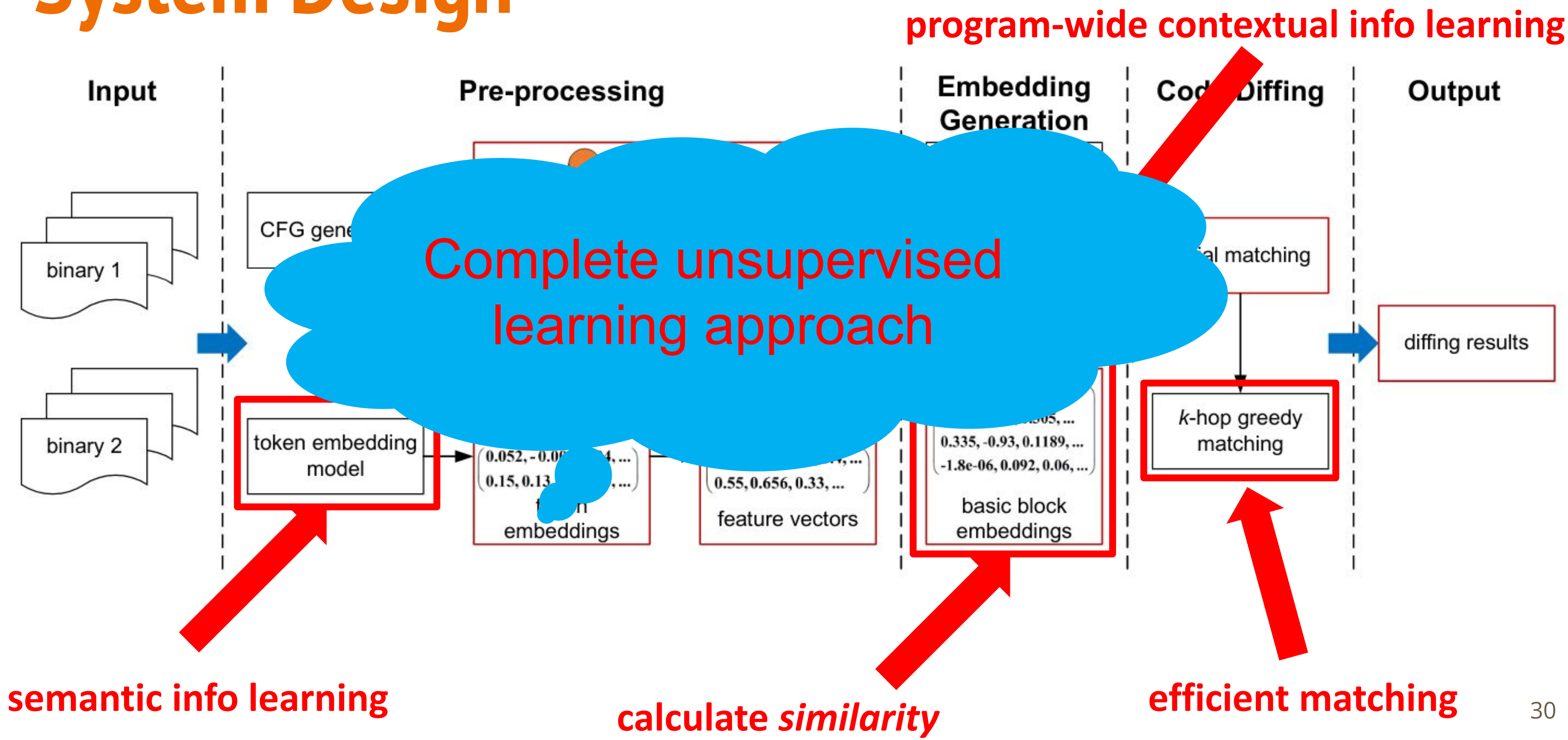
Yue Duan , Xuezixiang Li , Jinghan Wang , and Heng Yin

NDSS 2020

Motivation

- Existing techniques
  - No efficient binary diffing at basic block level
    - Genius and Gemini: function comparison
  - No program-wide contextual information
    - why useful?
  - Heavily rely on labelled training data
    - balanced training data can be hard

System Design



Key Idea: Natural Language Processing

binaries

A TEACHING GRAMMAR OF THE ENGLISH ARTICLE SYSTEM

Patricia L. McEldowney

Le professeur qui enseigne l'anglais comme langue étrangère a besoin de plus d'information sur l'expression des catégories grammaticales que celle qui est à sa disposition dans les grammaires descriptives de la langue anglaise. Il devrait pouvoir s'appuyer dans son enseignement sur une ‹Grammaire didactique› *(teaching grammar).* C'est surtout dans le domaine de l'emploi des articles que les étudiants de l'anglais comme langue étrangère commettent un grand nombre de fautes. Cet article essaie de développer une grammaire didactique du système des articles. En ce faisant, l'auteur part d'une description généralisée ayant pour but de présenter au professeur une vue d'ensemble, suivie d'information sur la fréquence d'occurrence des formes particulières et sur le caractère, optionnel ou obligatoire respectivement, de ces formes. De telles connaissances permettront au professeur de commencer par les règles fondamentales de l'usage quand il présente les matériaux en classe, et d'élargir ou, le cas échéant, de modifier successivement ces règles en vue d'un enseignement plus avancé.

Der Lehrer, der Englisch als Fremdsprache unterrichtet, braucht mehr Informationen über den Ausdruck grammatischer Kategorien, als er in deskriptiven Grammatiken der englischen Sprache finden kann. Sein Unterricht sollte sich auf eine ‚Didaktische Grammatik' *(teaching grammar)* stützen können. Im Bereich des Artikelgebrauchs werden von allen Lernenden des Englischen als Fremdsprache besonders viele Fehler gemacht. Dieser Aufsatz versucht, eine didaktische Grammatik des englischen Artikelsystems zu entwickeln. Dabei geht er von einer generalisierten Beschreibung aus, um dem Lehrer einen Gesamtüberblick zu geben. Dann folgen Informationen über die Gebrauchshäufigkeit der einzelnen Formen sowie darüber, welche Formen optional und welche obligatorisch verwendet werden. Derartige Kenntnisse befähigen den Lehrer bei der Darbietung des Stoffes im Unterricht, mit den grundlegenden Gebrauchsregeln anzufangen und diese im Fortgeschrittenenunterricht stufenweise zu erweitern bzw. zu modifizieren.

In any grammatical area the teacher of English to non-native speakers needs a great deal of information which he cannot easily find in an ordinary descriptive grammar of English. This type of grammar does not often contain, for instance, explicit statements about the usefulness of certain items, nor explicitly identify areas where items overlap in function. This indicates a need for what might be called "a teaching grammar" of English. Such a grammar would refashion the descriptive grammarian's organisation and insights to make explicit and readily available certain information of value to the teacher. This paper examines the teacher's grammatical needs with reference to article usage in English, and, in so doing, is a proposal for an outline of one section of such a teaching grammar.

In the paper, the phrase "article usage" refers to usage concerned with the presence or absence of the items *a, the, -s* and *some.*

random walks

Identify the subject and predicate in these SIMPLE sentences.

3. Cindy and Sue auditioned for the lead role in the play.

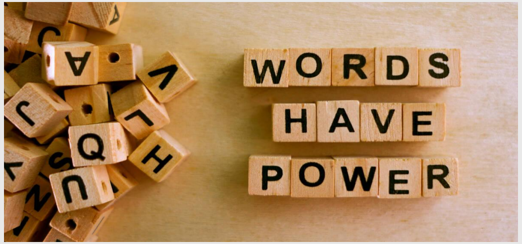
2. The kittens were adopted by the family.

3. Peanut butter and jelly sandwiches are my favorite.

4. The committee decorated the gym for Friday night's dance.

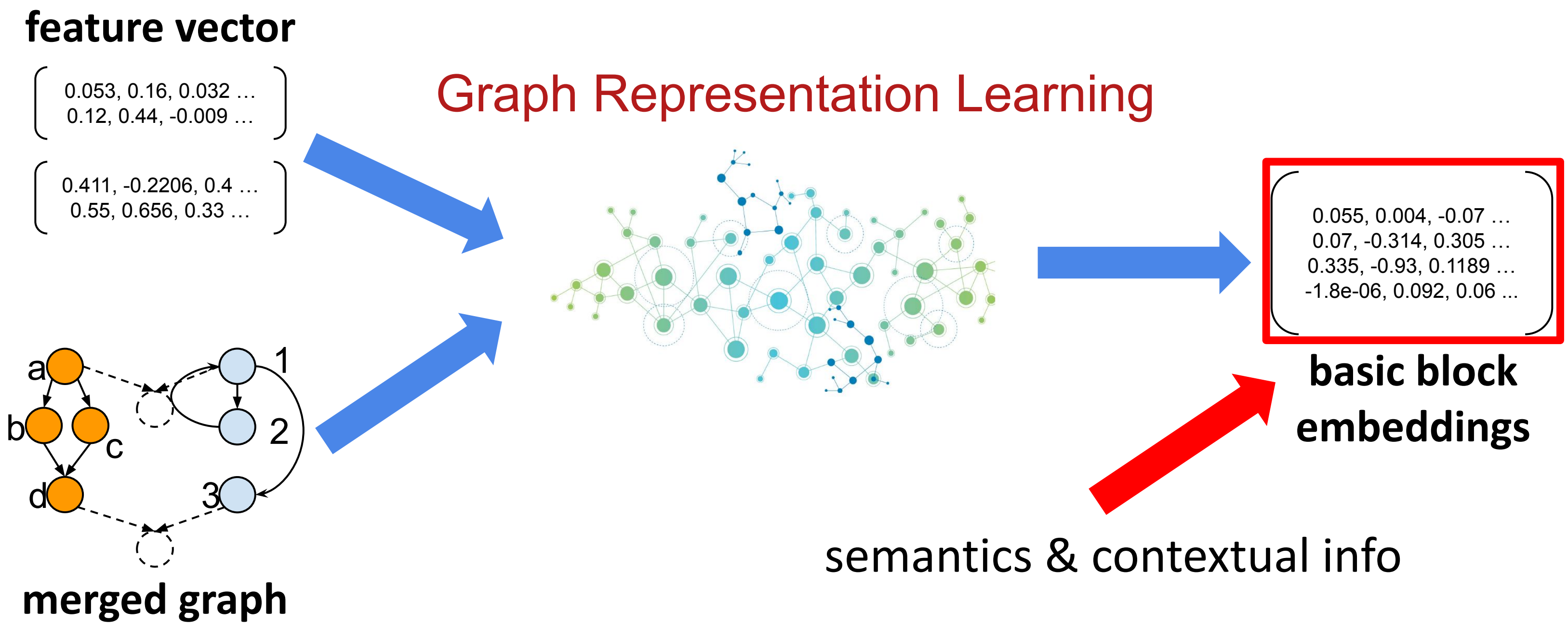
5. The surprise party was organized by Wendy's two best friends.

tokens



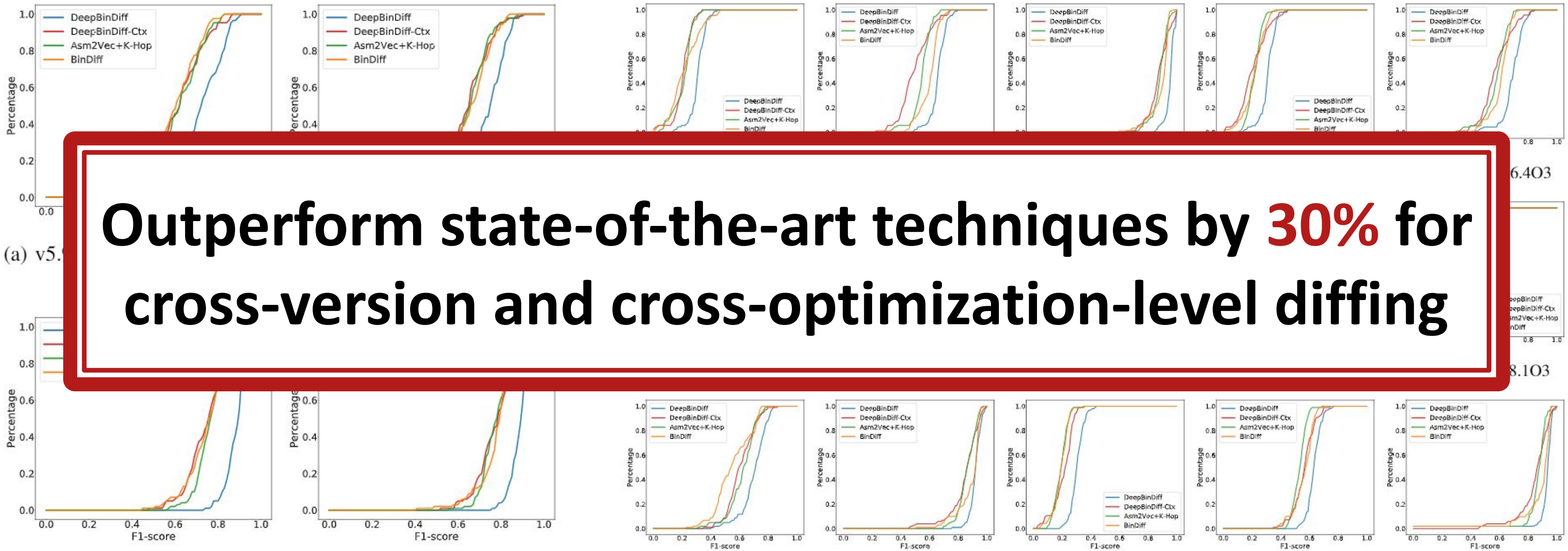
Program-wide Contextual Info Learning

Program-wide Contextual Info Learning



calculate basic block similarity

Evaluation



(a) v5.
6.4O3
8.1O3

(c) v7.6 compared with v8.30 (d) v8.1 compared with v8.30 (k) v8.1O1 vs v8.1O3 (l) v8.1O2 vs v8.1O3 (m) v8.30O0 vs v8.30O3 (n) v8.30O1 vs v8.30O3 (o) v8.30O2 vs v8.30O3

Outperform state-of-the-art techniques by 30% for cross-version and cross-optimization-level diffing

Thank you!

Question?