XDA: Accurate, Robust Disassembly with Transfer Learning

NDSS2021

Kexin Pei (Columbia University), Jonas Guan (University of Toronto),

David Williams-King (Columbia University), Junfeng Yang (Columbia University), Suman Jana (Columbia University)

-21.03.02-

김 정 우

목차

I. INTRODUCTION

II. BACKGROUND

III. OVERVIEW OF OUR APPROACH

IV. THREAT MODEL

V. METHODOLOGY

VI. IMPLEMENTATION AND EXPERIMENTAL SETUP

VII. EVALUATION

VIII. CASE STUDIES

IX. RELATED WORK

X. CONCLUSION

Introduction

- Disassembly to recover assembly instruction and function
  - Difficult
  - High-level information are absent in stripped binaries

- Traditional approaches rely on hand-crafted heuristics
  - IDA Pro, Ghidra's have their own databases for identifying function

Introduction

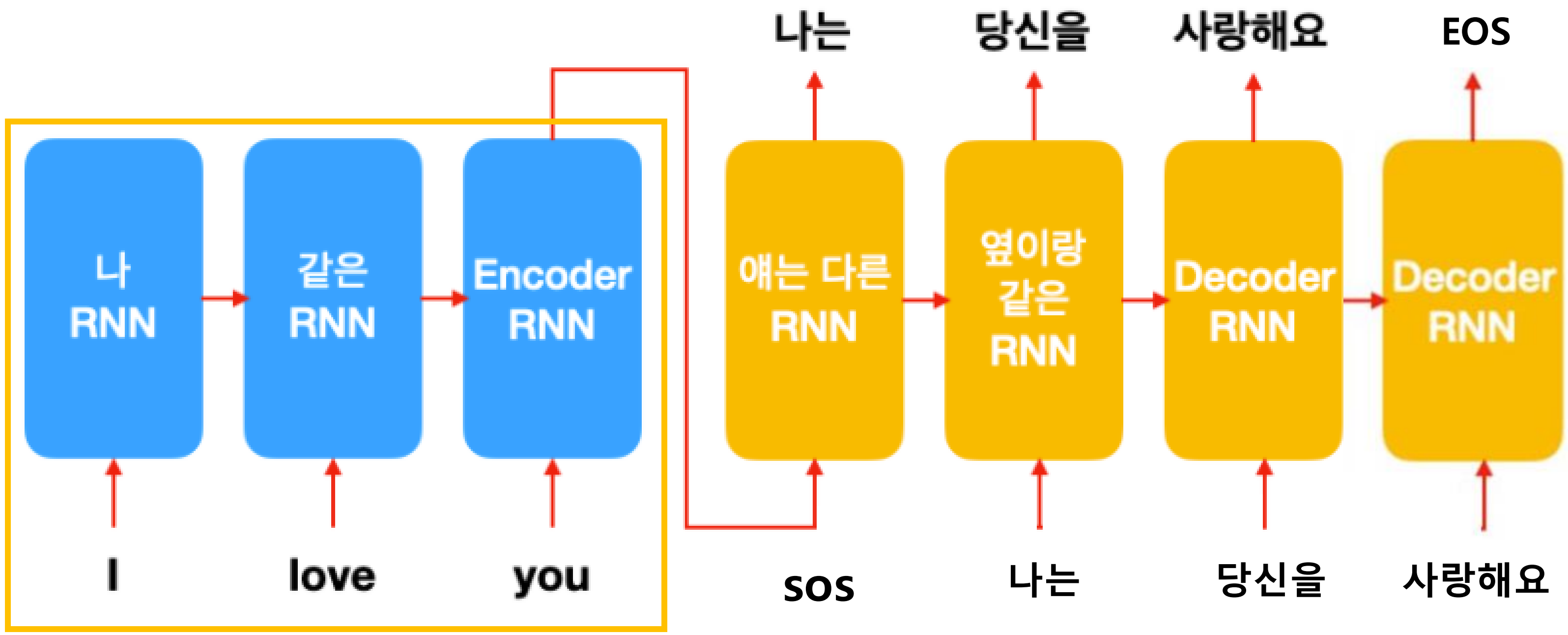
- ML-based Method for disassemble
  - Those approaches face two challenges
    1. Accuracy
    2. Robustness : not robust to compiler optimization

- XDA (Xfer-learning DisAssembler)
  - ML-based disassembly framework that address these challenges.
    1. outperforms all state-of-the-art tools
    2. robust to changes in compiler optimization
    3. XDA's speed is on par with the fastest tools

XDA

바이너리를 Assembly로 바꾸는 번역기

Sequence-to-Sequence : 번역



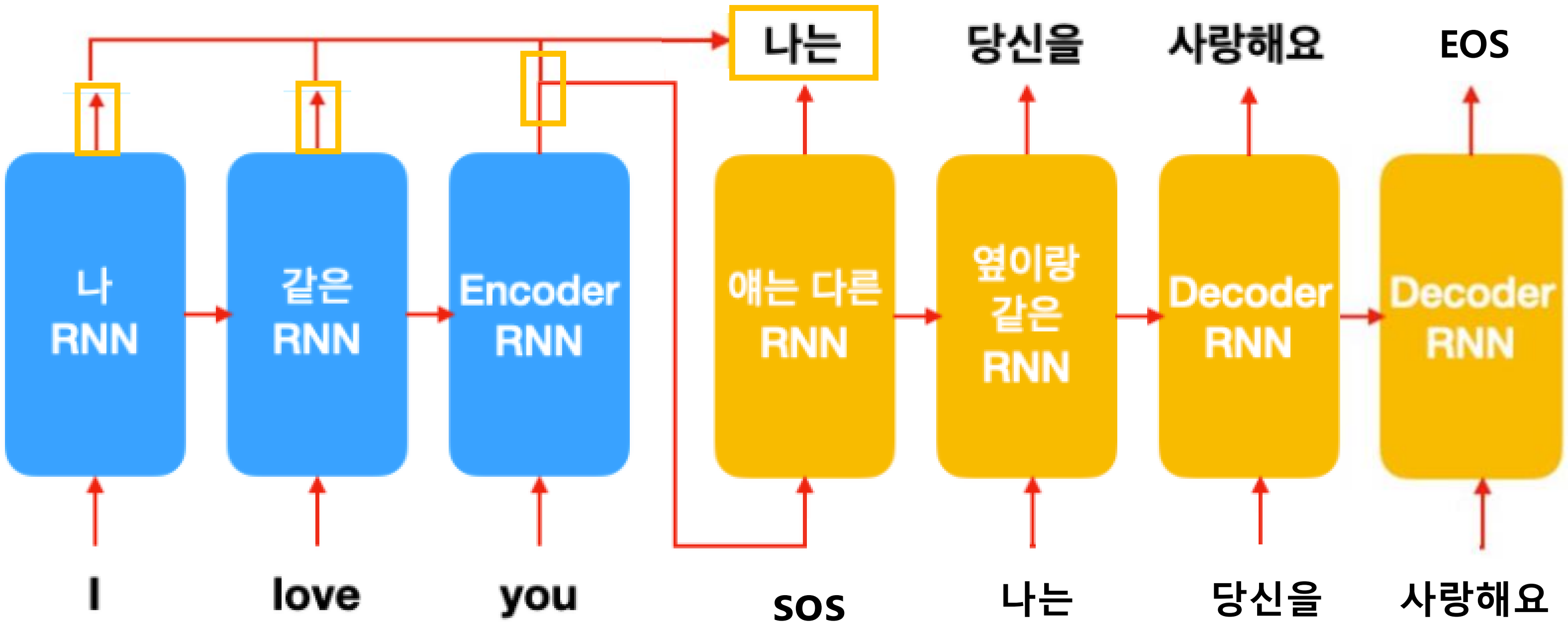
영어 언어 모델

seq2seq의 기본 구조

Sequence-to-Sequence

- I love you -> 나는 당신을 사랑해요
  - 기존 Seq2Seq 로 구현시 정보 손실 문제
  - 결국 문장을 **순차적으로 이해**시키는 것은 정보 손실을 피할 수 없음
  - 이를 보정하기 위해 Attention!

Sequence-to-Sequence + Attention

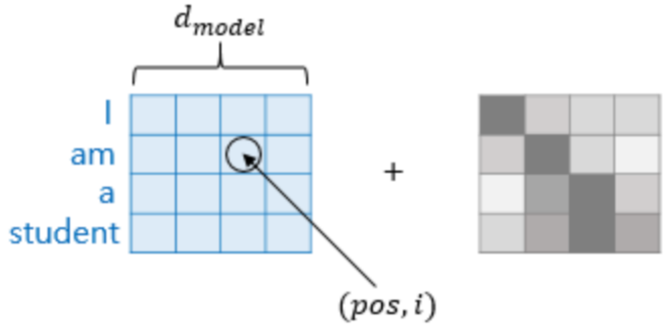


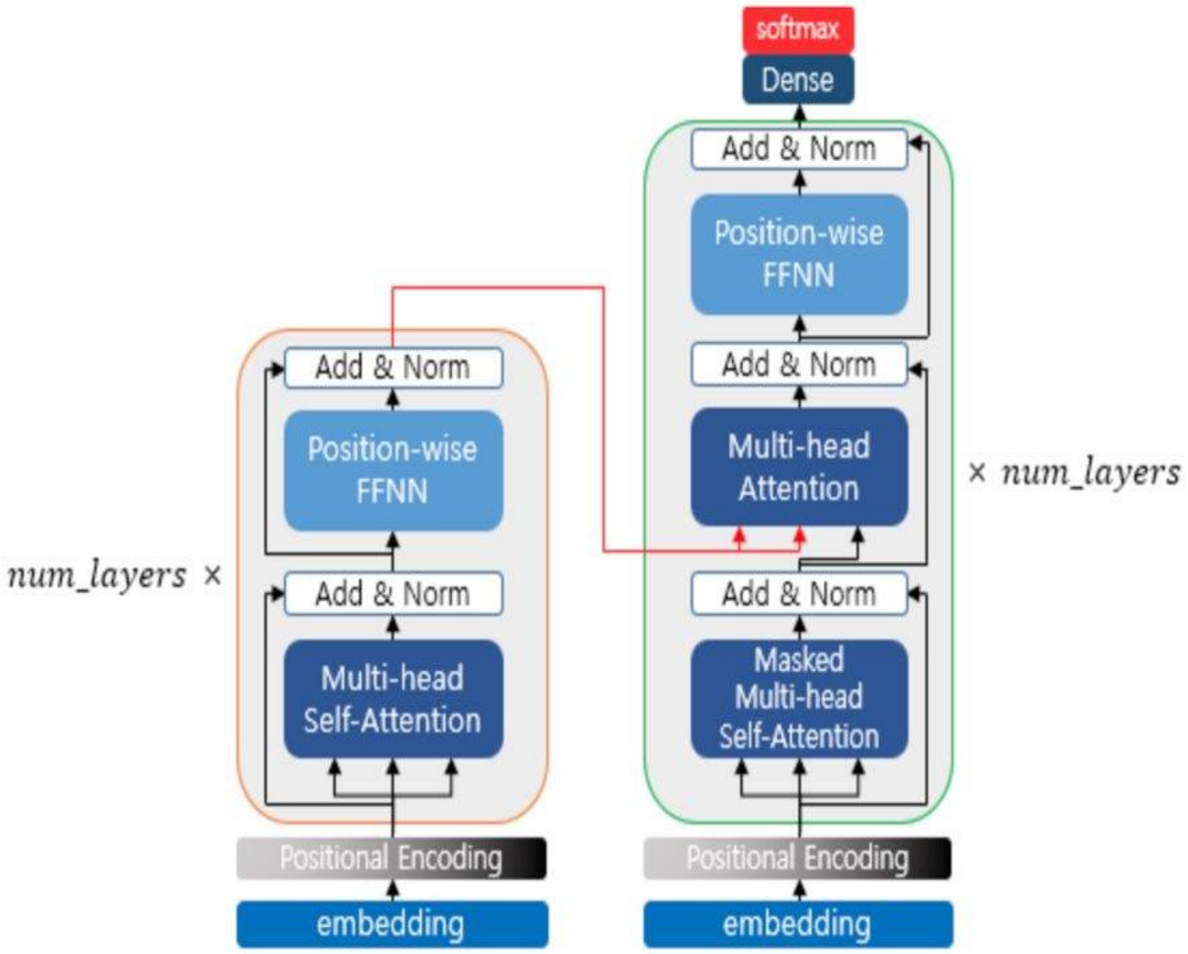
seq2seq의 기본 구조

Transformer

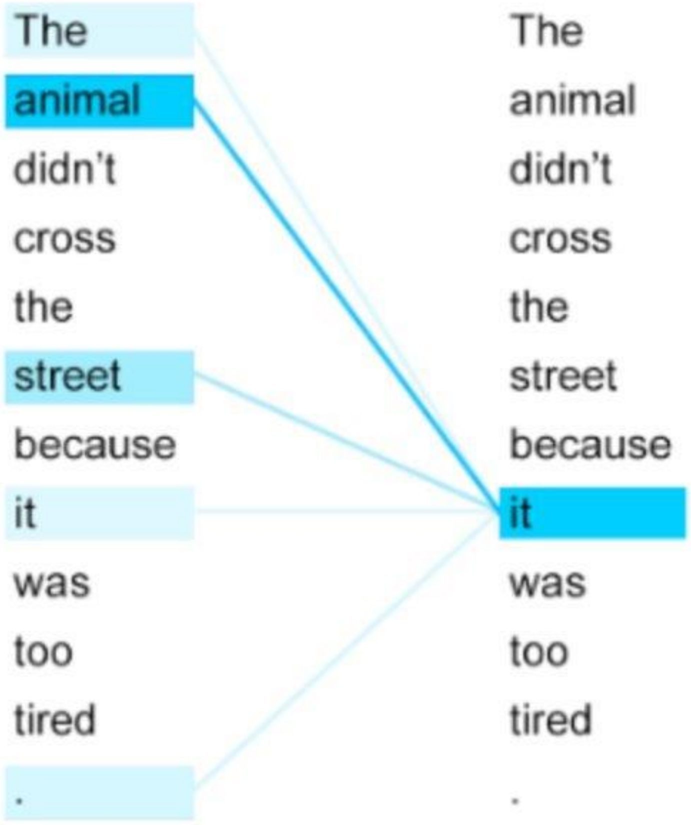
Attention만으로 인코더와 디코더를 만들자

- 학습을 위한 **Self-Attention**
  - **+ Positional encoding**

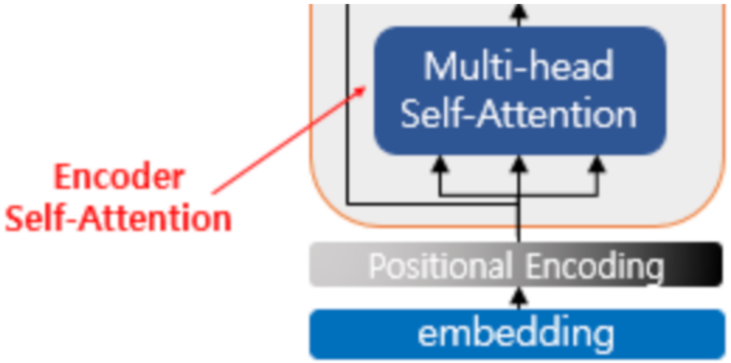




Transformer : Self-Attention



- Multi-head Attention
  - 좌측과 같은 연산을 병렬처리

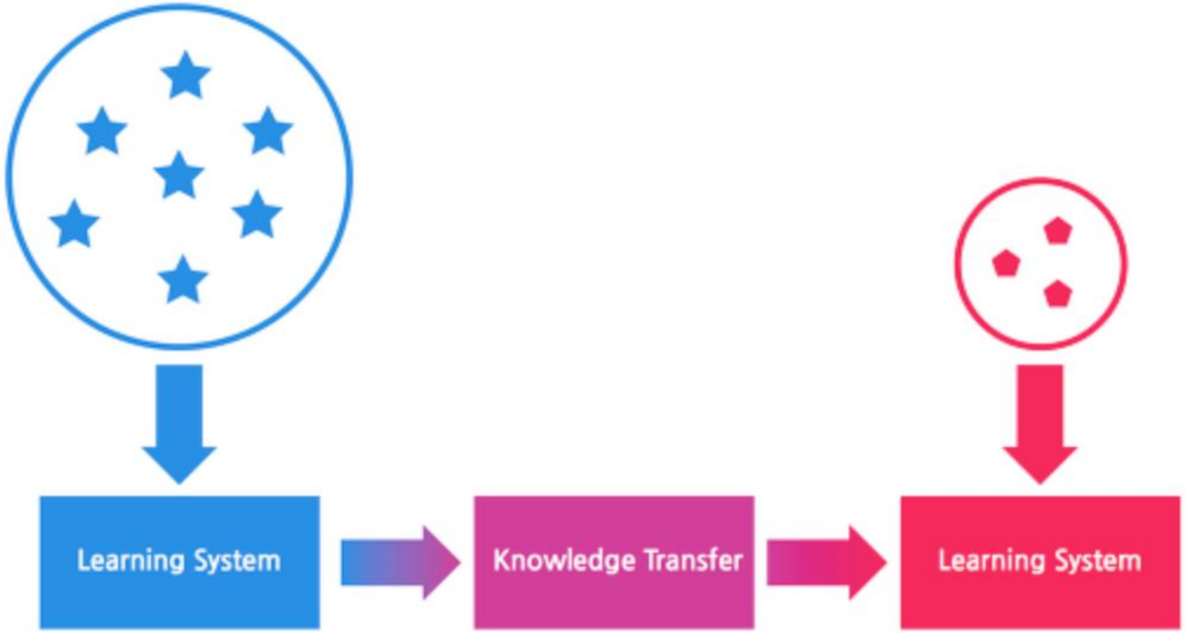


Pretrained

Transfer Learning

Pretrained Model and Fine tuning

- 언어를 인공신경망에 학습시키는 것은 매우 큰 작업
- 미리 방대한 양의 데이터를 학습시키고 학습된 결과를 가져오면 어떨까?
- 이를 위한 전이학습



| 전이학습

Language Model

- $P(wn|w1,...,wn-1)$ = N번째 단어에 대한 확률

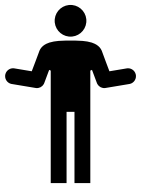
- $P(W)=P(w1,w2,w3,w4,w5,...,wn)$
  하나의 단어를 w, 단어 시퀀스을 대문자 W

- 언어모델을 RNN과 같은 신경망으로 구현

- Language model을 표현할 수 있다면, 다양한 언어 태스크를 처리가능
  - Ex) 번역기

나는 학생입니다.

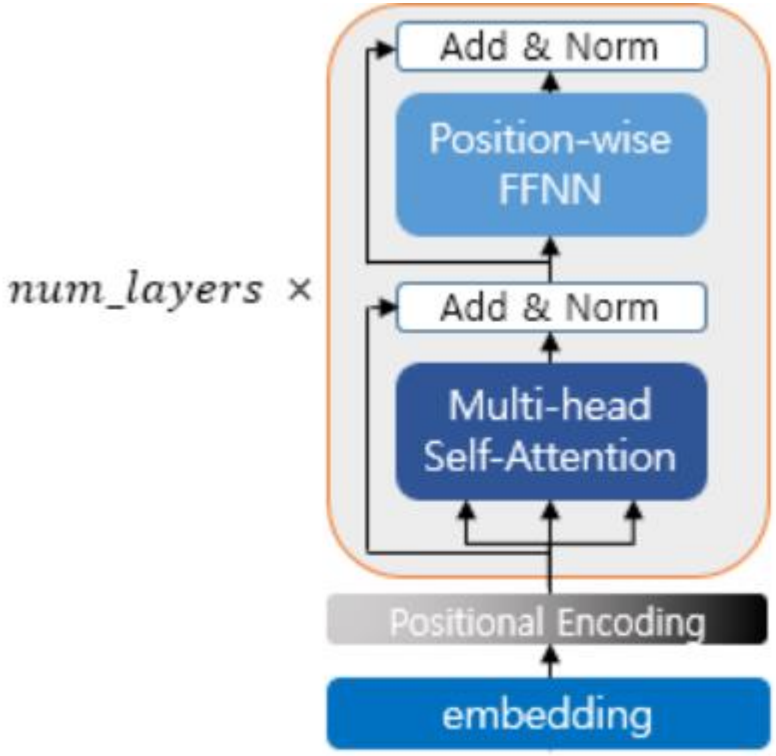
"나" + "는" + "학생" + "입니다".

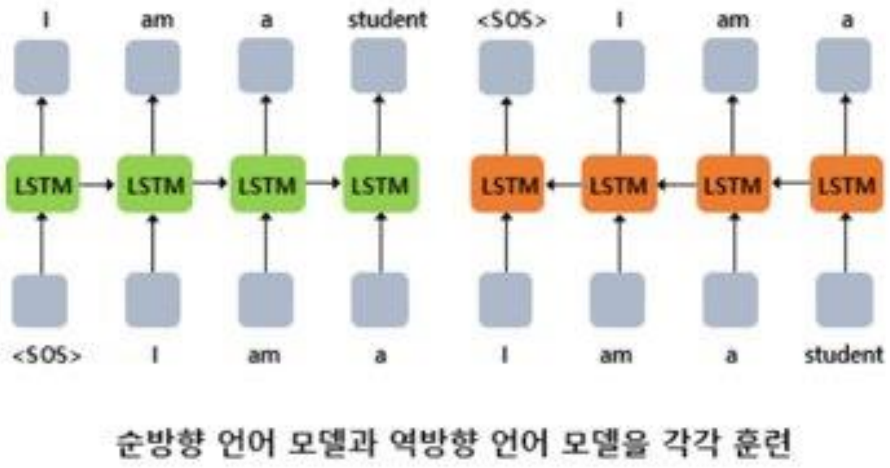


나 > 는 > 학생 > 입니다.

나 < 는 < 학생 < 입니다.

Language Model : limitaion of Transformer



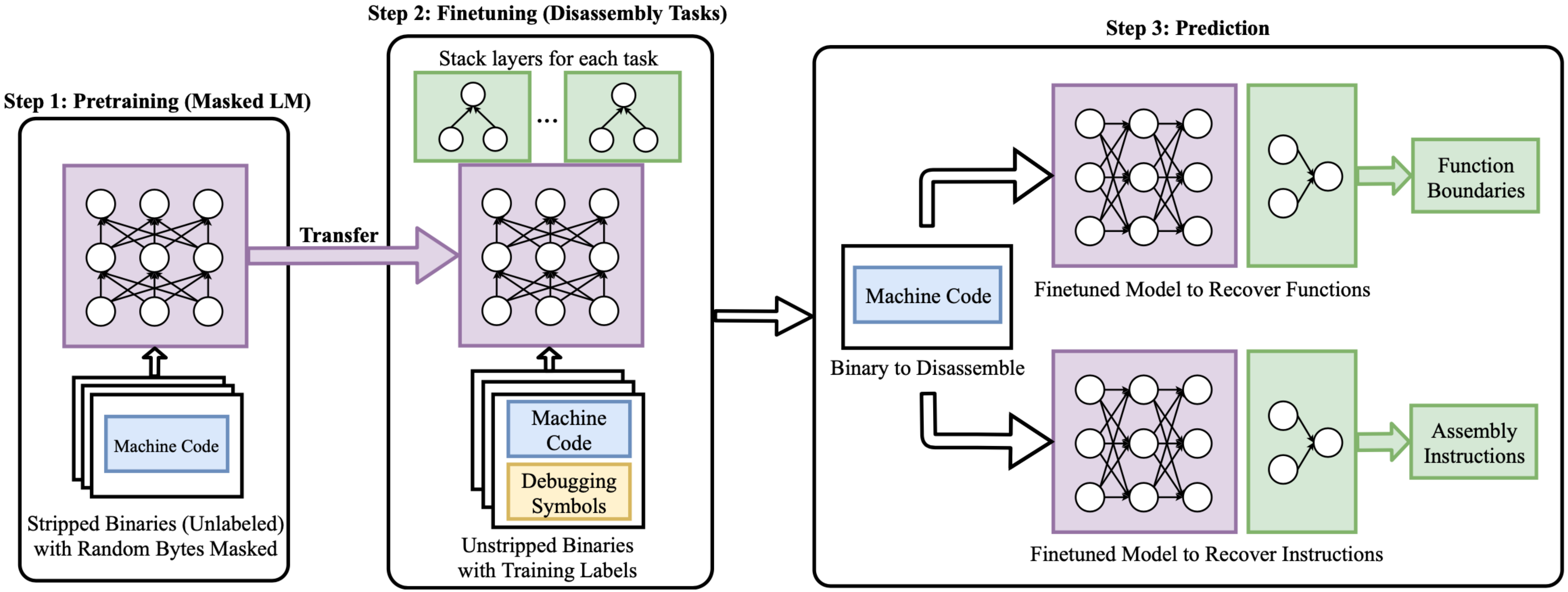


인공신경망의 훈련방식

Masked Language Model

- 입력 텍스트의 15%의 단어를 랜덤으로 마스킹(Masking)
- 인공 신경망에게 이 가려진 단어들을(Masked words) 예측
- Ex) The man went to the store → The man went to the [MASK]

The workflow of XDA



THREAT MODEL

- Robustness.
  - Do not aim to be robust in the presence of arbitrarily **obfuscated code**
  - Aim to be robust against **compiler changes**

- False positives and negatives
  - Assume a small number of false positives and negatives can be tolerated.

How to pretrain XDA for dissambly

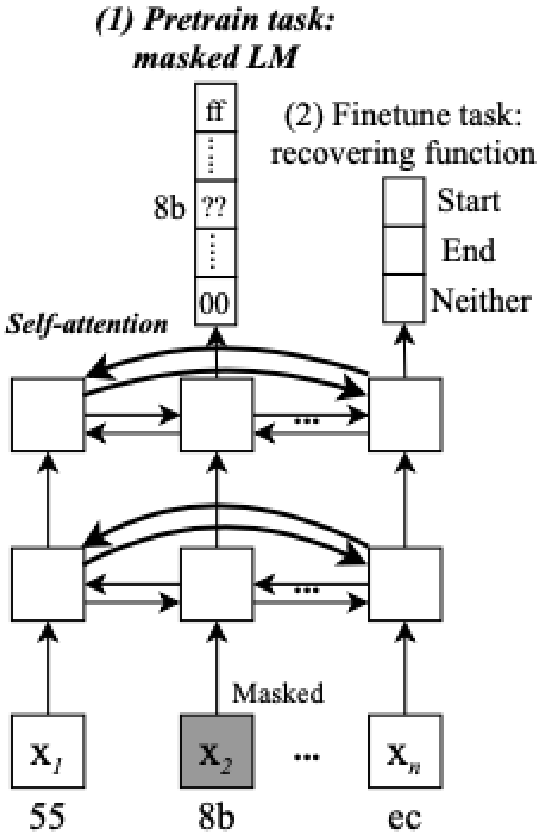
- Byte position embeddings
  - learned positional embedding Epos

- *Masked Language Model on Binaries*
  1. choose 20% random bytes to mask
  2. select 50% of them to be replaced by the special token <MASK>
  3. replace with random bytes in the vocabulary {0x00, ..., 0xff}

METHODOLOGY

- Multi-head self-attention.

- Contextualized embeddings



(a) XDA architecture

IMPLEMENTATION AND EXPERIMENTAL SETUP

Dataset	Total # Binaries	Platform	Compiler	ISA	# Binaries	# Bytes	# Byte Sequences	Train-test Overlap
SPEC 2017	588	Linux	GCC-9.2	x86	120	198,019,576	386,757	0.001%
				x64	224	464,906,401	908,021	0.2%
		Windows	MSVC-2019	x86	88	175,057,814	341,910	0.93%
				x64	156	955,201,152	1,865,628	0.97%
SPEC 2006	333	Linux	GCC-5.1.1	x86	90	55,637,428	108,667	0.002%
				x64	95	74,006,029	144,543	0%
		Windows	MSVC-2019	x86	76	40,417,016	78,940	0.36%
				x64	72	48,403,456	94,538	0.21%
BAP	2,200	Linux	GCC-4.7.2 & ICC-14.0.1	x86	1,032	138,547,936	270,602	1%
				x64	1,032	145,544,012	284,266	1.1%
		Windows	MSVC-2010 & 2012 & 2013	x86	68	29,093,888	56,824	0.4%
				x64	68	33,351,168	65,139	2.3%

Pretrained on SPEC CPU2006 and BAP , finetuned with SPEC CPU2017

IMPLEMENTATION AND EXPERIMENTAL SETUP

- Baselines
  - IDA Pro v7.4
  - Ghidra v9.1
  - Objdump
  - Nucleus, which is based on the control-flow analysis
  - bi-RNN to recover function boundaries

IMPLEMENTATION AND EXPERIMENTAL SETUP

- Label collection for fine tuning
  - Debug symbols and source code of the binaries
    - Windows binaries, we parse PDB files using Dia2dump
    - Linux binaries, we parse DWARF information using the pyelftools

- Metrics
  - Precision, Recall, F1-score
  - Perplexity : (문장을 헷갈리는 정도 낮을 수록 좋음)
  - Train-test overlap rate : (훈련,테스트 데이터의 바이너리 시퀀스 중첩)

EVALUATION

- **RQ1: How accurate is XDA in recovering function boundaries and instructions compared to other tools?**
- **RQ2: How robust is XDA under different platforms, compilers, architectures, and optimizations, compared to other tools?**
- RQ3: How fast is XDA compared to other tools?
- RQ4: How efficient is XDA in terms of saving labeling effort and training epochs compared to other tools?
- RQ5: How effective is pretraining, and how does it help finetuning tasks?

EVALUATION : RQ1

Dataset	Platform	ISA	Recovering Function Boundaries F1 (%) XDA	Nucleus	bi-RNN	IDA	Ghidra	Recovering Instructions F1 (%) XDA	bi-RNN	IDA	Ghidra	objdump
SPEC 2017	Linux	x86	**98.4**	55.4	79.9	91.8	89.0	99.9	87.1	95.9	94.6	**100.0**†
		x64	**99.1**	55.0	79.2	90.2	89.5	99.9	88.9	95.8	95.9	**100.0**†
	Windows	x86	**99.1**	60.8	73.8	67.6	70.4	99.2	82.3	96.7	92.1	**99.3**
		x64	**98.9**	65.0	78.4	78.0	71.6	**99.4**	81.9	97.1	93.1	99.3
SPEC 2006	Linux	x86	**98.2**	57.2	86.7	95.7	92.2	99.9	89.0	96.3	95.5	**100.0**†
		x64	**98.7**	56.8	73.8	92.8	92.0	99.8	85.9	96.4	94.9	**100.0**†
	Windows	x86	**99.4**	68.2	78.5	77.9	76.3	**99.7**	89.9	98.1	94.5	99.1
		x64	**98.3**	56.8	72.7	90.1	86.2	**99.4**	86.2	97.9	95.7	99.4
BAP	Linux	x86	**99.5**	61.5	74.1	59.0	57.2	N/A*	N/A*	N/A*	N/A*	N/A*
		x64	**98.7**	53.5	79.0	58.3	56.5	N/A*	N/A*	N/A*	N/A*	N/A*
	Windows	x86	**99.5**	69.0	80.1	89.9	87.0	N/A*	N/A*	N/A*	N/A*	N/A*
		x64	**99.4**	70.0	81.4	90.5	80.6	N/A*	N/A*	N/A*	N/A*	N/A*
Average			**99.0**	60.8	78.1	81.8	79.0	**99.7**	86.4	96.8	94.4	99.6

Tasks of recovering function boundaries and assembly instructions.

EVALUATION : RQ1

- Generalizability

TABLE III. F1 SCORE (%) OF XDA AND BI-RNN ON RECOVERING FUNCTION BOUNDARIES ON VARYING TRAIN-TEST OVERLAP RATE.

	Train-test Overlap Rate			
	20%	40%	60%	80%
bi-RNN	70.1	82.3	89.8	96.5
XDA	99.1	99.5	99.8	99.9

EVALUATION : RQ1

- Generalizability to real-world

TABLE IV. F1 SCORE OF XDA'S FUNCTION BOUNDARY RECOVERY (PRETRAINED ON BAP AND SPEC CPU2006 AND FINETUNED ON SPEC CPU2017 X64 BINARIES COMPILED ON LINUX WITH GCC) ON UNSEEN BINARIES COLLECTED FROM POPULAR OPENSOURCE PROJECTS.

	O0	O1	O2	O3
Curl	98.6	98.5	98.6	98.2
Diffutils	98.7	98.7	98.8	98.6
GMP	99.2	98.8	98.9	98.6
ImageMagick	98.4	98.3	98.2	98.2
Libmicrohttpd	98.9	98.8	98.9	98.7
LibTomCrypt	99.0	99.0	98.7	98.6
OpenSSL	98.4	98.3	98.4	98.2
PuTTy	98.3	98.3	98.2	98.1
SQLite	98.8	98.7	98.4	98.3
Zlib	98.9	98.9	99.0	98.8

EVALUATION : RQ1

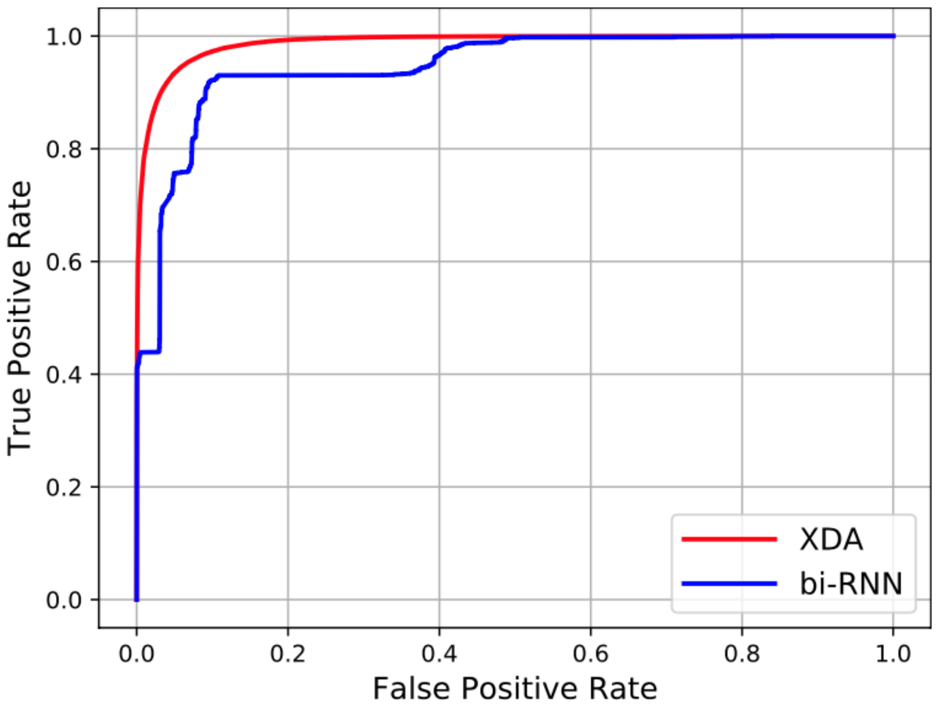


Fig. 6. The ROC curve of XDA and bi-RNN, when finetuning and testing on SPEC CPU2017 Windows x64 binaries compiled by MSVC.

EVALUATION : RQ2

- Robustness to different optimizations.

TABLE V. TEST F1 SCORE (%) OF XDA AND BI-RNN TRAINED AND TESTED ON DIFFERENT OPTIMIZATION FLAGS.

	Train OPT \ Test OPT	O1	O2	Od	Ox
bi-RNN	O1	**81**	80	47	2.8
	O2	44	**85**	81	75
	Od	34.5	4.1	**85.2**	43.6
	Ox	80	39	44	**87**
XDA	O1	**99.8**	98.6	98.8	98.5
	O2	99.8	**98.7**	99	98.9
	Od	99.6	98.5	**99**	98.7
	Ox	99.7	98.7	98.8	**98.9**

EVALUATION : RQ2

TABLE VI. F1 SCORE OF XDA'S FUNCTION BOUNDARY RECOVERY ON UNSEEN BINARIES OBFUSCATED WITH 5 DIFFERENT OBFUSCATION TYPES.

	bcf	cff	ibr	spl	sub
Curl	98.3	98.5	98.6	98.5	99.0
Diffutils	98.6	98.7	98.4	98.7	99.1
GMP	99.0	98.9	98.9	98.5	99.2
ImageMagick	98.3	98.3	98.1	98.0	98.4
Libmicrohttpd	98.6	98.7	98.8	98.6	98.9
LibTomCrypt	98.7	98.6	98.7	98.6	98.9
OpenSSL	98.3	98.9	98.6	98.9	99.0
PuTTy	98.2	98.1	98.1	98.0	98.3
SQLite	98.7	98.6	98.1	98.4	98.8
Zlib	98.0	98.4	98.1	98.6	99.0

CONCLUSION

- XDA′s potential for a wide range of downstream disassembly and binary analysis tasks
- We open-source XDA at https://github.com/CUMLSec/XDA.