

# 캡스톤 디자인 I

## 종합설계 프로젝트

프로젝트 명	asi(a security insight)
팀 명	assist(a security safety important special team)
문서 제목	2차 중간보고서

Version	1.2
Date	2020-05-25

팀원	손 현기 (조장)
	김 주환
	김 호준
	오 예린
	이 동윤
	RUSLAN
지도교수	윤 명근 교수

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24

### CONFIDENTIALITY/SECURITY WARNING

이 문서에 포함되어 있는 정보는 국민대학교 전자정보통신대학 컴퓨터공학부 및 컴퓨터공학부 개설 교과목 캡스톤 디자인I 수강 학생 중 프로젝트 “asi(a security insight)”를 수행하는 팀 “assist(a security safety important special team)”의 팀원들의 자산입니다. 국민대학교 컴퓨터공학부 및 팀 “assist(a security safety important special team)”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다.

## 문서 정보 / 수정 내역

<b>Filename</b>	2차중간보고서-asi(a security insight).doc
<b>원안작성자</b>	김주환
<b>수정작업자</b>	김주환

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2020-05-18	김주환	1.0	최초 작성	초안 작성
2020-05-24	김주환	1.1	추가 작성	수행 결과 추가
2020-05-25	김주환	1.2	수정 작성	오타 수정

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24

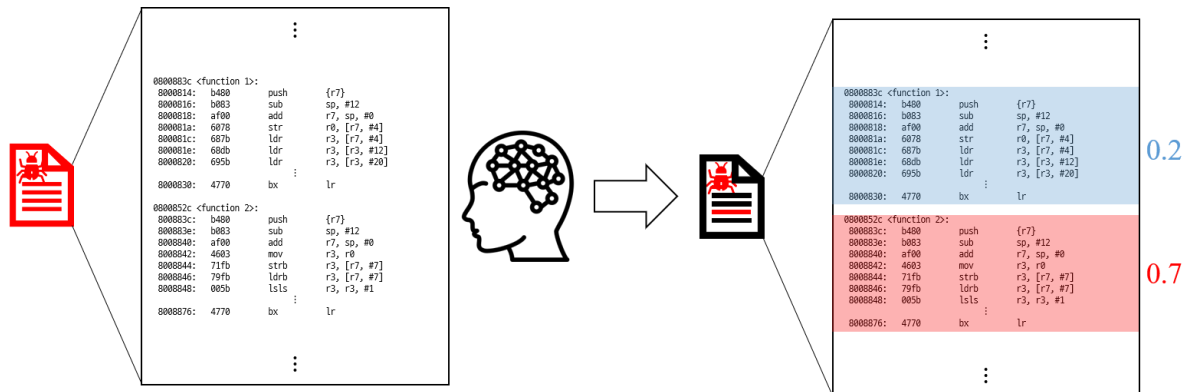
## 목 차

1	프로젝트 목표 .....	4
2	수행 내용 및 중간결과 .....	6
2.1	계획서 상의 연구내용.....	6
2.1.1	데이터 수집 .....	7
2.1.2	딥러닝 기반 특징벡터 추출 도구.....	7
2.1.3	딥러닝 기반 악성행위 추출 도구.....	8
2.1.4	웹 서비스 .....	10
2.1.5	수행 계획 .....	11
2.2	수행내용 .....	11
2.2.1	데이터 수집 .....	11
2.2.2	데이터 전처리 .....	12
2.2.3	딥러닝 기반 특징벡터 추출 도구.....	14
2.2.4	딥러닝 기반 악성행위 추출 도구.....	20
2.2.5	웹 서비스 .....	27
3	수정된 연구내용 및 추진 방향.....	30
3.1	수정사항 .....	30
4	향후 추진계획 .....	31
4.1	향후 계획의 세부 내용.....	31
4.1.1	딥러닝 기반 악성행위 추출 도구.....	31
4.1.2	웹 서비스 .....	31
5	고충 및 건의사항 .....	32
6	부록 .....	33

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24

## 1 프로젝트 목표

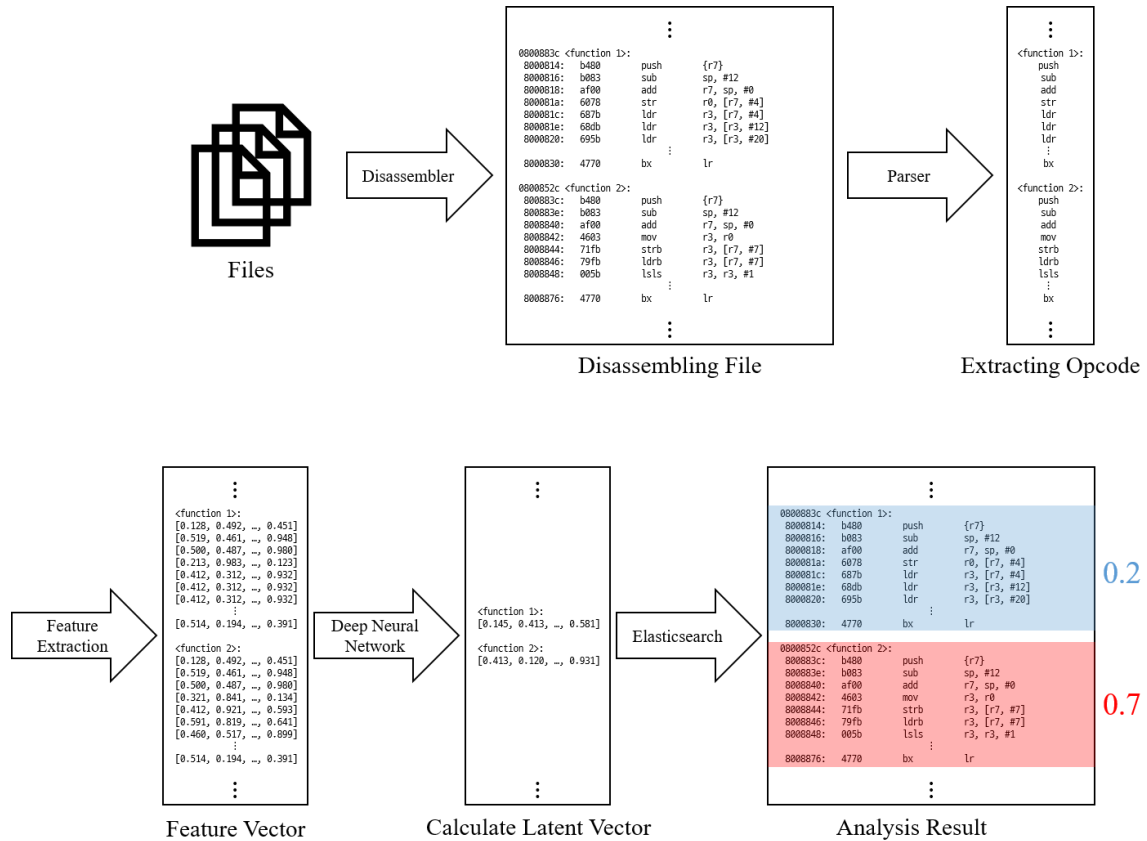
본 프로젝트의 목표는 [그림 1]과 같이 실행파일의 각 함수를 분석하여 악성 행위를 수행할 것으로 예측되는 정도를 점수화 하는 딥러닝 기반 소프트웨어를 개발함으로써 전문가의 악성코드 분석을 돕는 것이다.



[그림 1] 제안하는 소프트웨어의 목표

구현하고자 하는 소프트웨어의 구체적인 흐름은 [그림 2]와 같다. 분석하고자 하는 파일이 입력되면 역어셈블러(disassembler)를 이용하여 어셈블리 코드를 추출하고, 추출한 어셈블리 코드 중 연산자(Op Code, Operation code)를 파서를 이용해 얻는다. 신경망을 이용해 학습 및 분류를 하기 위해서는 입력을 특징벡터(feature vector)로 변환하는 과정이 필요하므로, 단어 임베딩(word embedding)과 같은 임베딩 알고리즘을 이용하여 추출한 연산자를 적합한 특징공간의 특징벡터로 변환한다. 신경망은 특징벡터로부터 압축벡터(latent vector)를 추출한다. 마지막으로 Elasticsearch를 이용해 데이터베이스에서 입력 파일과 유사한 함수를 찾는다. 유사한 함수들 중 악성행위를 수행하는 함수의 비중을 기반으로 입력 함수의 악성 행위 가능성을 점수화한다.

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24



[그림 2] 제안하는 소프트웨어의 분석 방법 개요

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24


## 2 수행 내용 및 중간결과

### 2.1 계획서 상의 연구내용

본 프로젝트의 세부 목표는 [표 1]과 같다. 1 단계는 신경망 학습을 위한 데이터를 수집 및 전처리하는 단계이다. 2 단계는 순환신경망(Recurrent Neural Network, RNN) 학습을 위해 역어셈블링한 결과를 적합한 특징공간의 특징벡터로 변환하는 단어 임베딩 방법론을 개발하는 단계이다. 3 단계는 본 프로젝트의 목표인 악성행위 추출을 위한 순환신경망 기반의 신경망을 구현 및 학습시키고, 이를 사용자가 편리하게 사용하기 위해 웹 서비스를 구축하는 단계이다. [표 2]는 이상의 목표에 대응하는 결과물 목록을 나타낸 것이다.

[표 1] 프로젝트의 세부 목표

<b>1 단계 연구 목표</b>	
	<ul style="list-style-type: none"> <li>- 정상파일 및 악성파일 데이터셋 확보</li> <li>- 니모닉 추출 알고리즘 개발</li> </ul>
<b>2 단계 연구 목표</b>	
	<ul style="list-style-type: none"> <li>- 딥러닝 기반 특징벡터 추출 도구 개발</li> </ul>
<b>3 단계 연구 목표</b>	
	<ul style="list-style-type: none"> <li>- 딥러닝 기반 이상탐지 알고리즘 확보</li> <li>- 딥러닝 기반 자동 악성행위 코드 추출 도구 개발</li> <li>- 악성코드 분석 보조도구 제공을 위한 서버 구축</li> </ul>

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24

[표 2] 프로젝트의 세부 결과물

<b>1 단계 결과물</b>	<ul style="list-style-type: none"> <li>- 정상파일을 수집하는 크롤러</li> <li>- 어셈블리 코드로부터 연산자와 피연산자를 추출하는 파서</li> <li>- 정상파일 및 악성파일 데이터셋</li> </ul>
<b>2 단계 결과물</b>	<ul style="list-style-type: none"> <li>- 단어 임베딩 소프트웨어 (Word2Vec, FastText, Doc2Vec)</li> <li>- 1 단계의 데이터셋에 대한 특징벡터</li> </ul>
<b>3 단계 결과물</b>	<ul style="list-style-type: none"> <li>- 각 하이퍼 파라미터별 실험 결과</li> <li>- 최적화된 신경망 모델</li> <li>- 자동 악성행위 코드 추출 도구</li> <li>- 분석 파일 업로드 및 결과 시각화를 위한 웹</li> </ul>

### 2.1.1 데이터 수집

신경망을 적절히 학습하기 위해서는 양질의 데이터가 필요하다. 우리는 학습에 필요한 정상 파일과 악성파일을 얻기 위해 Microsoft Kaggle 프로젝트 malware prediction의 데이터셋, 한국인터넷진흥원의 정보보호 R&D 데이터 챌린지의 데이터셋 등을 이용한다. 추가로 충분히 많은 정상파일을 얻기 위해 크롤러(crawler)를 직접 개발한다.

우리는 수집한 정상 및 악성파일로부터 어셈블리 코드를 추출하기 위해 상용 역어셈블러 프로그램인 IDA Pro를 사용한다. 본 프로젝트에서는 어셈블리 코드를 이용하여 악성행위를 탐지하므로 Python을 기반으로 어셈블리 코드에서 연산자와 피연산자를 추출하는 파서를 개발한다.

### 2.1.2 딥러닝 기반 특징벡터 추출 도구

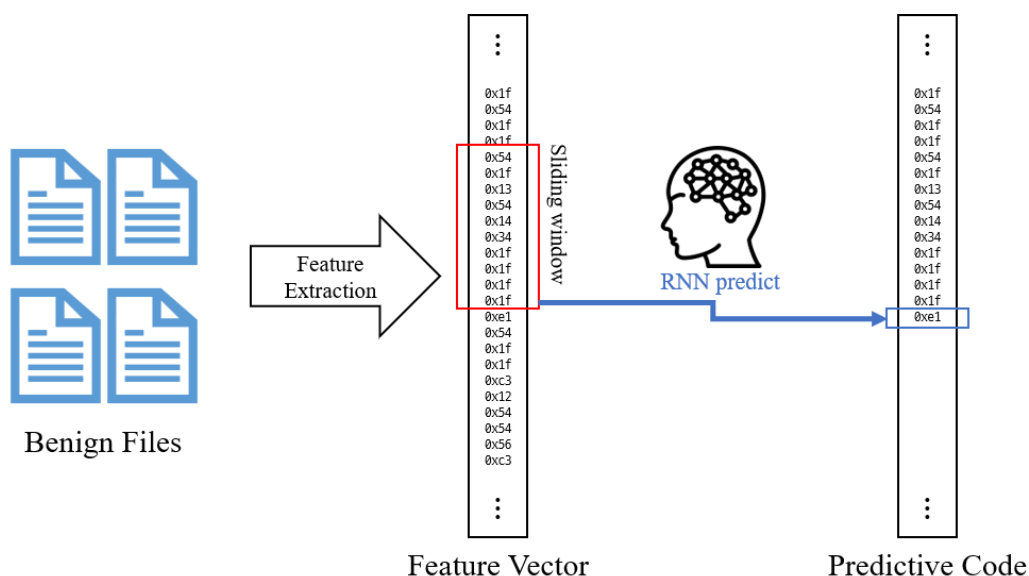
학습을 위해서는 추출한 연산자와 피연산자를 적합한 특징공간의 특징벡터로 변환하는 과정이 필요하다. 이를 위해 Word2Vec, Doc2Vec, FastText와 같은 딥러닝 기반 단어 임베딩 알고리즘을

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24

적용한다. 또한, 변환된 결과를 시각화 함으로써 적합한 특징벡터 추출 방법론을 선정한다.

### 2.1.3 딥러닝 기반 악성행위 추출 도구

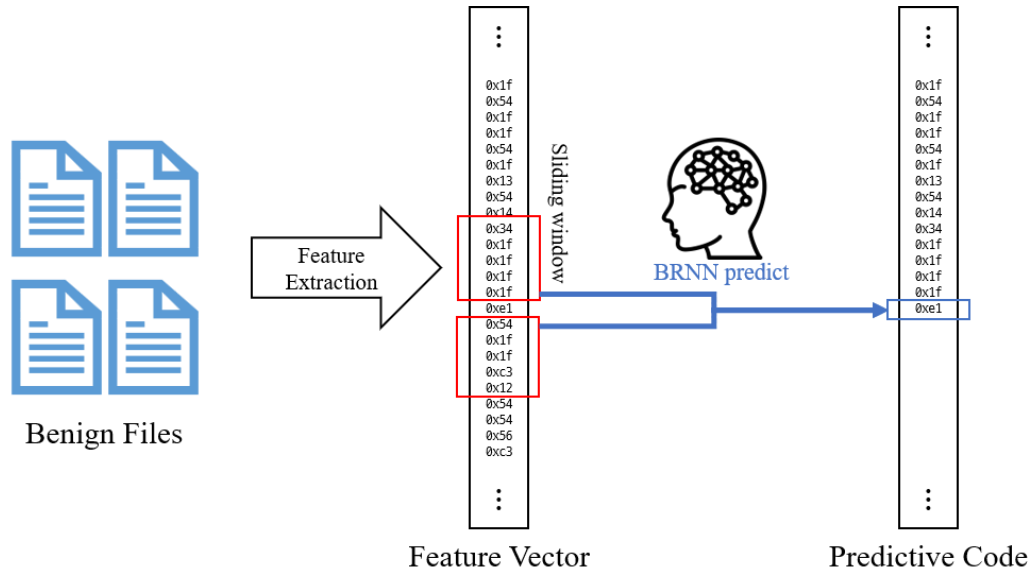
우리는 RNN을 기반으로 정상파일 연산자의 언어모델을 학습시킴으로써 악성파일 중 악성행위를 수행하는 부분을 추출한다. 학습단계에서는 정상파일을 이용하여, 이전 연산자로 다음 연산자를 예측하는 방식과 주변 연산자로 중심 연산자를 추측하는 방식, 두 가지 방식을 실험하고, 최적의 신경망을 채택한다. 이때 사용하는 RNN의 구조로는 Vanilla RNN, LSTM(long short-term memory models), GRU(gated recurrent unit)를 사용한다.



[그림 3] 이전 연산자로 다음 연산자를 예측하는 학습 방안

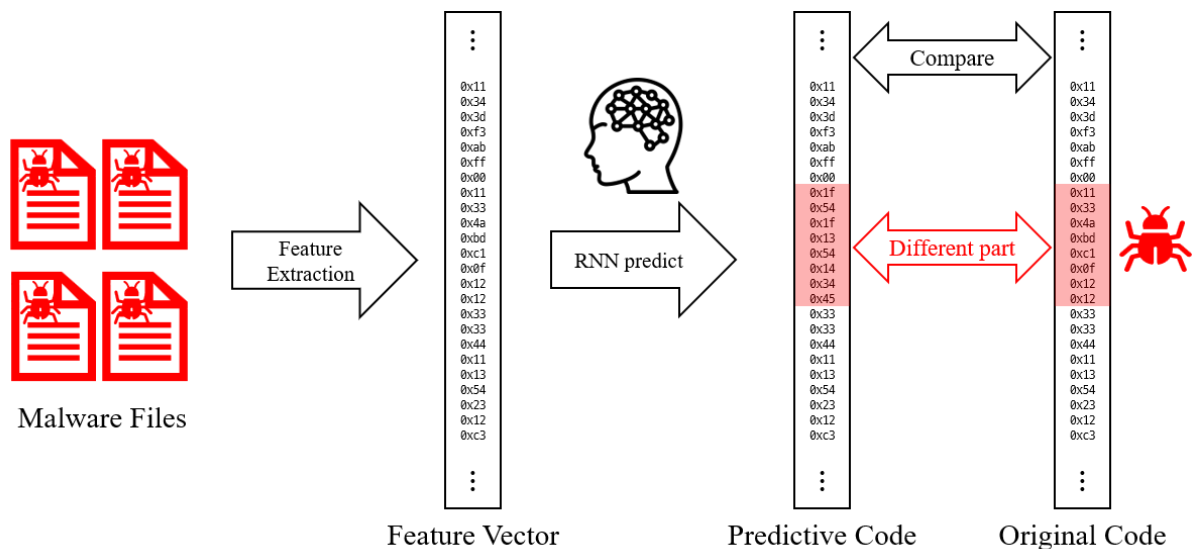


 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24



[그림 4] 주변 연산자로 중심 연산자를 예측하는 학습 방안

악성행위 수행부를 추출하는 방법은 [그림 5]와 같다. 학습된 신경망이 예측한 연산자와 원본 연산자를 비교했을 때 신경망이 제대로 예측하지 못하는 지점을 악성행위를 수행하는 부분이라 추정한다.

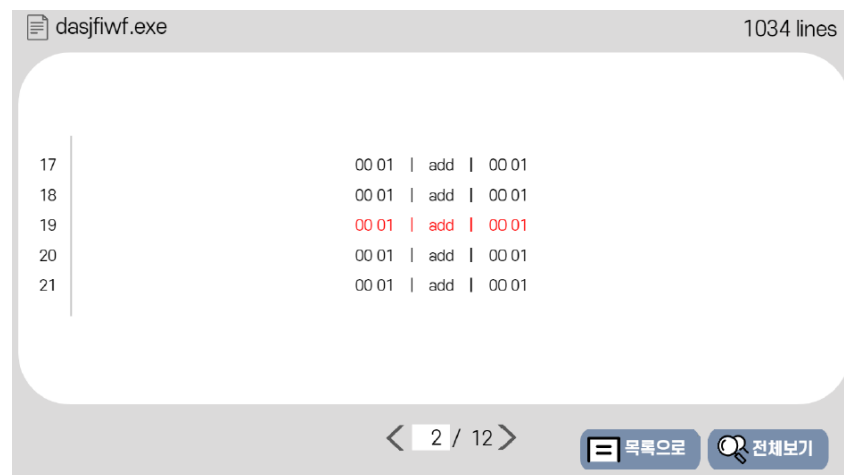


[그림 5] 악성행위 코드 추출 방안

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24

## 2.1.4 웹 서비스

제안된 시스템을 사용자 친화적으로 제공하기 위하여 웹을 기반으로 서비스를 제공한다. 웹은 사용자가 악성으로 의심되는 파일을 업로드하면 [그림 6]과 같이 코드를 분석한 결과를 시각화하여 나타낸다.



[그림 6] 웹 기반 분석 결과 시각화 안

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24

### 2.1.5 수행 계획

[표 3] 프로젝트 수행 계획

항목	세부내용	1 월	2 월	3 월	4 월	5 월	6 월
요구사항분석	요구 분석						
	SRS 작성						
관련분야연구	딥러닝 기술 연구						
	관련 논문 동향조사						
구현	웹 크롤러 제작 및 데이터 수집						
	파서 구현						
	단어 임베딩						
	신경망 구현						
	웹 서버 구축						
	웹 프론트엔드 개발						
테스트	시스템 테스트						

## 2.2 수행내용

### 2.2.1 데이터 수집

학습에 필요한 정상 및 악성파일을 수집하기 위하여 Microsoft의 Kaggle 프로젝트인 Microsoft malware classification<sup>1</sup>의 데이터셋, 한국인터넷진흥원의 정보보호 R&D 데이터 챌린지의 데이터셋을 활용한다. Microsoft malware classification 프로젝트는 악성파일의 패밀리를 분류하는 것을 목표로 한다. 데이터셋은 바이트 파일 10,868개와 어셈블리 파일 10,868개로 총 21,738개이며, 각 악성파일의 패밀리가 제공된다. 한국인터넷진흥원 정보보호 R&D 데이터 챌린지<sup>2</sup>는 2017년부터 시행된 경진대회이며 정상파일과 악성파일을 분류하는 것을 목표로 한다. 데이터셋은 정상파일 10,000

<sup>1</sup> <https://www.kaggle.com/c/malware-classification/overview>

<sup>2</sup> <http://datachallenge.kr/>

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24

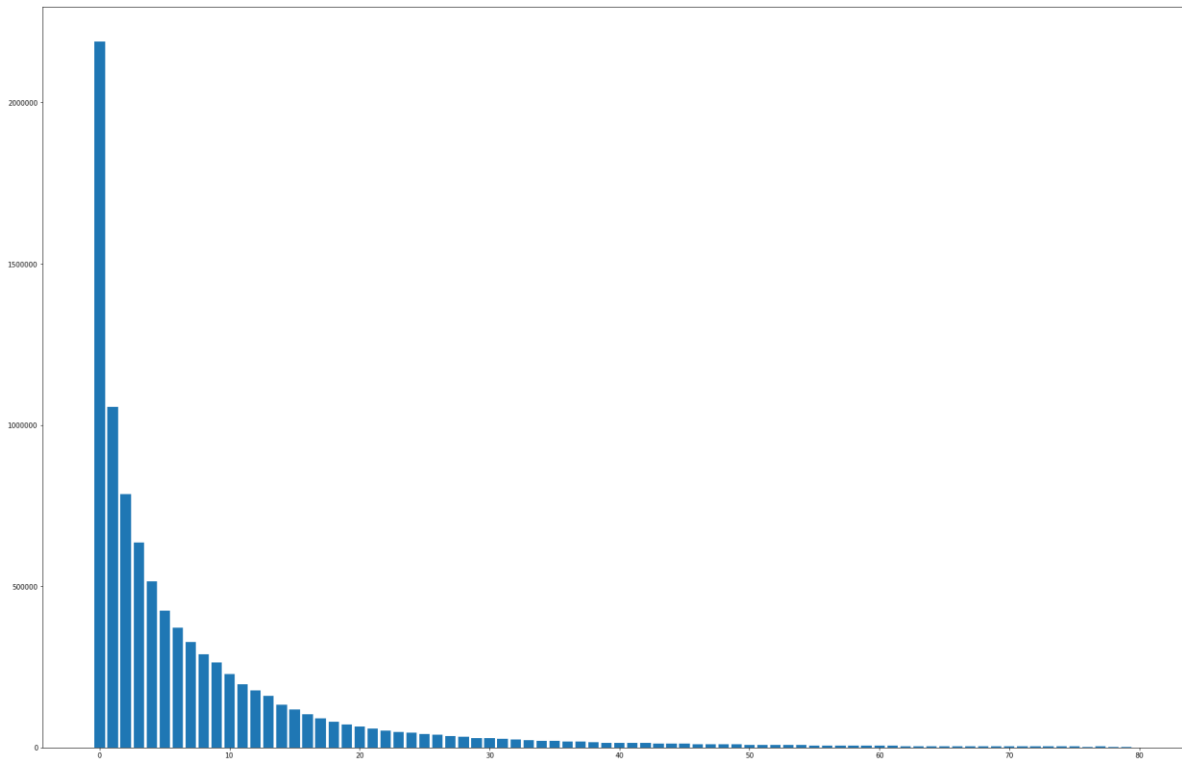
개와 악성파일 10,000개이며, 각 파일의 정상/악성 여부가 제공된다.

본 프로젝트에서는 학습을 위해 정상파일을 사용하므로 양질의 정상파일을 확보해야 한다. 이를 위해 우리는 정상파일을 수집하기 위한 크롤러를 제작했다. 크롤링 대상은 높은 신뢰도로 정상이라 판정할 수 있는 시스템의 DLL(dynamic link library) 파일과 Steam 사의 게임 인스톨러로 지정했다. 각 크롤러는 GitHub의 crawler 폴더에서 확인할 수 있다.

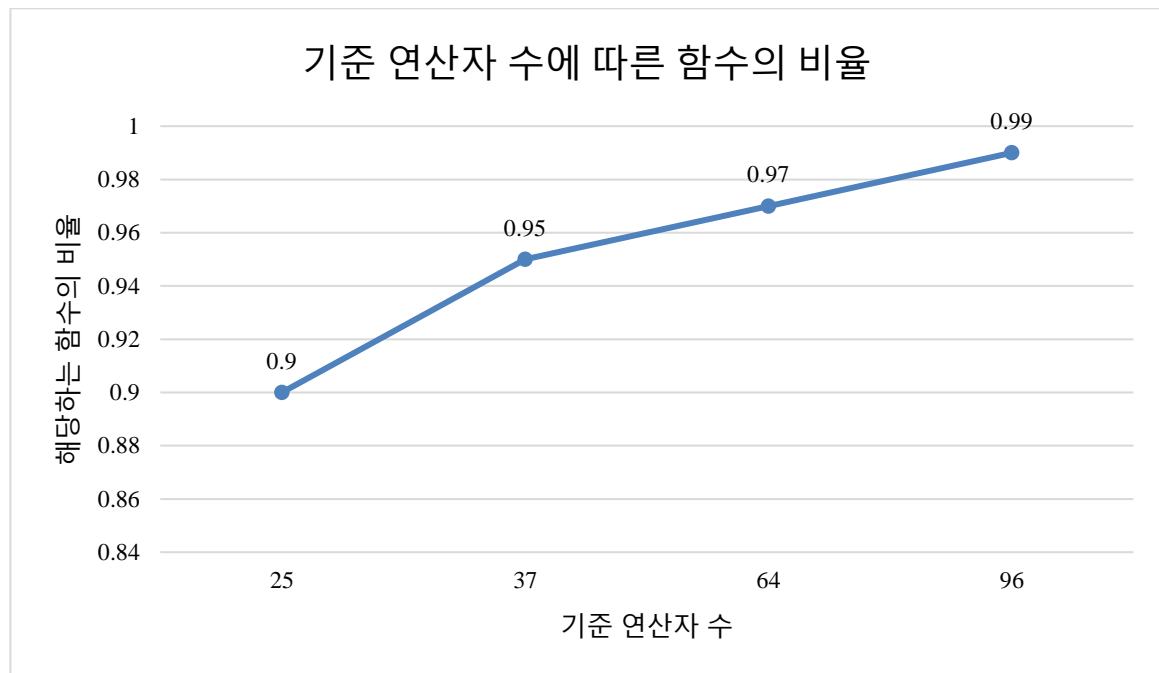
### 2.2.2 데이터 전처리

수집한 데이터셋은 실행파일과 DLL 파일이므로, 학습을 위해서는 파일 중 어셈블리 코드를 추출하는 과정이 필요하다. 우리는 이를 위해 상용 역어셈블러 프로그램인 IDA Pro를 사용했으며, 역어셈블된 결과 중 연산자를 추출하기 위해 파서를 Python으로 직접 구현했다. 구현한 파서는 IDA Pro 상에서 동작하며, 역어셈블된 결과를 기본 블록(basic block) 단위로 나눈 것을 Python 피클(pickle)로 저장하도록 구현했다.

제안된 신경망은 오토인코더로 구성되므로 입력 벡터의 크기를 동일하게 구성하는 것이 필요하다. 즉, 함수의 연산자의 개수를 일정하게 맞추는 작업이 필요하다. 이때 연산자의 개수가 너무 작으면 너무 많은 정보를 잃게 되며, 개수가 너무 크면 신경망의 너비가 넓어져 과적합이 발생할 수 있으므로 적합한 기준 연산자 수를 찾아야 한다. 적합한 기준을 찾기 위해 우리는 5,000개의 정상 파일의 각 함수 별 연산자의 수를 분석했다. [그림 7]은 이를 도식화한 것으로, x축은 연산자의 수, y축은 함수의 수를 의미한다. 분석 결과 짧은 함수가 대부분의 비중을 차지하며, 연산자의 수가 증가할수록 해당하는 함수의 수는 감소함을 볼 수 있다. 이를 기반으로 적합한 연산자의 수를 구하기 위해 100,000개의 정상 파일에 대해 기준 연산자의 수에 따른 함수의 비율을 나타낸 것이 [그림 8]과 같다. 그림에서 연산자의 수가 64개 이하인 함수의 비율은 97%로 대부분의 함수를 포함하므로 기준 연산자 수는 64로 지정했다. 입력 함수의 연산자의 수가 기준 연산자 수보다 작다면, 남은 부분을 “padding”이라는 단어로 패딩하였으며, 이보다 길다면 앞 64개의 연산자만 활용하도록 구성하였다.



[그림 7] 각 함수별 연산자의 수



[그림 8] 연산자 수에 따른 함수의 비율

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24

또한, 명령어의 수가 너무 많아져 학습에 방해가 되는 것을 방지하기 위해 파생되는 명령어를 하나의 명령어로 통합하였다. 예를 들어 mov.ovf는 mov로, push.ovf는 push로 통합하였다.

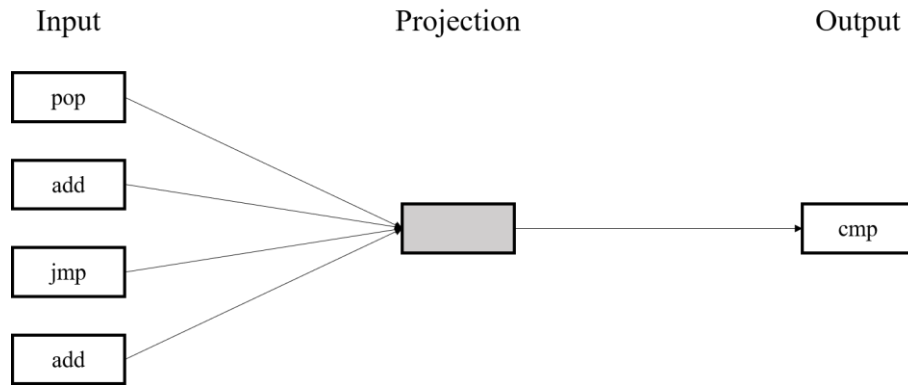
### 2.2.3 딥러닝 기반 특징벡터 추출 도구

#### 가) 관련 연구

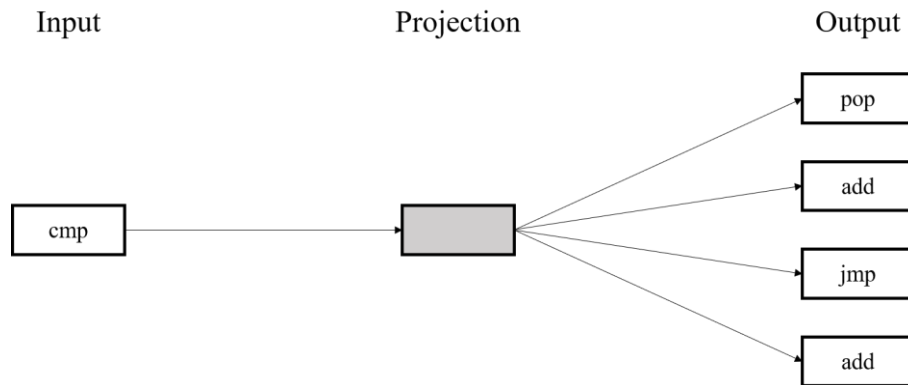
RNN을 적절히 학습시키기 위해서는 연산자를 적합한 특징벡터로 변환시키는 과정, 즉 단어 임베딩 과정을 적절히 수행해야 한다. 우리는 이를 위해 딥러닝 기반의 Word2Vec, Doc2Vec, FastText 알고리즘을 적용하여 임베딩 신경망을 구성한다.

Word2Vec은 가장 많이 사용되는 임베딩 방법론 중 하나로, 기존 원 핫 인코딩(one-hot encoding) 방식과 다르게 단어의 개수보다 낮은 차원의 벡터로 단어를 임베딩할 수 있으며, 유사한 단어를 가까운 공간에 사상(mapping)시키는 방식이다. 예를 들어 사과, 배, 서울 세 단어를 임베딩한다면 원핫 인코딩은 단어간의 관계를 고려하지 않고 일괄적으로  $(001)_2, (010)_2, (100)_2$ 로 임베딩하지만, Word2Vec은 사과와 배는 과일이라는 동일한 범주에 있으므로 가까운 공간에, 서울은 먼 공간에 사상시키는 방식이다. Word2Vec은 주변 단어를 이용해 중간 단어를 예측하는 방식인 CBOW(continuous bag of words) 방식과 중간 단어로 주변 단어를 예측하는 방식인 Skip-Gram 방식이 있다. 예를 들어 연산자가 pop, add, cmp, jmp, add이고, 윈도우 사이즈가 2이며, cmp 명령어에 대해 학습하는 단계라면 CBOW는 [그림 9]와 같이 주변 연산자인 pop, add, jmp, add를 입력 받아 cmp를 예측하도록 학습하며, Skip-Gram은 [그림 10]과 같이 중심 연산자인 cmp를 입력 받아 주변 연산자인 pop, add, jmp, add를 예측하도록 학습한다. 일반적으로 Skip-Gram 방식이 CBOW 방식보다 효과적이라 알려져 있으며, 우리의 실험 결과 역시 Skip-gram이 CBOW에 비해 더 효과적으로 임베딩을 수행함을 확인했다.

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24



[그림 9] CBOW 학습 방안



[그림 10] Skip-Gram 학습 방안

Word2Vec 학습시 고려해야 할 주요 하이퍼 파라미터는 출력 벡터의 차원, 윈도우 크기, 최소 단어의 수, 에포크(epoch), 학습률(learning rate)이다. 출력 벡터의 차원은 임베딩 결과로 나오는 벡터의 차원을 의미하고, 윈도우 크기는 학습에 사용하는 주변 단어의 개수를 의미한다. 위의 그림은 윈도우 크기가 2인 경우이다. 최소 단어의 수는 등장 빈도가 적은 단어를 임베딩 시키지 않기 위해 필요하며, 이 숫자보다 단어가 적게 등장하는 경우 임베딩 시키지 않는다. 에포크와 학습률은 일반적인 딥러닝에서의 의미와 동일하다. 에포크는 전체 데이터의 학습을 반복하는 횟수를, 학습률은 오류 역전파(back propagation)시 가중치를 변화시키는 비중을 의미한다.

Doc2Vec은 Word2Vec의 변형으로 학습의 입력으로 단어뿐만 아니라 문서의 색인을 추가로 받는다. Doc2Vec은 각 문서의 단어가 가까운 특징공간에 사상되는 것을 목적으로 한다. Word2Vec과 유사하게 주변 단어를 입력 받아 중심 단어를 예측하는 DBOW(distributed bag of words) 방식과 중심 단어를 입력 받아 주변 단어를 예측하는 Distributed-Memory 방식이 있다.

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24

FastText는 Word2Vec의 변형으로 단어의 부분이 일치하는 단어는 유사한 단어임을 이용한다. 이를 위해 단어를 작은 단위로 나누는 방법론을 채택한다. 예를 들어 imul이라는 연산자가 있고, 단어를 나누는 단위가 3이라면, imu, mul으로 나누어진 단어와 전체 단어 imul을 임베딩한다. 이후 mul이라는 연산자를 임베딩할 때는 imul과 가까운 특징 공간으로 사상될 수 있다. FastText의 학습 방법은 Word2Vec과 동일하다.

### 나) [실험 결과 1] 적합한 임베딩 알고리즘 탐색

우리는 gensim 라이브러리<sup>3</sup>를 이용하여 이상의 세 알고리즘을 구현했다. 하이퍼 파라미터는 디폴트 파라미터를 참고하여 윈도우 크기 10, 최소 단어 수 50, 에포크 10, 학습률 0.0002로 지정했다. RNN 구현시 모델의 너비를 적게 유지하면서 단어를 충분히 적합한 특징 공간으로 사상시키기 위해 우리는 특징벡터의 차원을 8, 16, 32, 64, 128로 각 단어의 의미를 충분히 나타내는 최소 차원을 찾는 것을 목적으로 실험했으며, 이때 학습 데이터는 정상 파일 5,000개를 활용했다.

단어가 적합한 특징 공간으로 사상 되었는지 확인하기 위해 우리는 다섯 개의 연산자 mov, jmp, add, pop, push에 대해 가장 가까운 위치에 사상된 연산자를 확인했으며, 사상시킨 결과를 t-SNE로 시각화 했다.

실험 결과 Word2Vec 알고리즘이 임베딩에 가장 적합함을 확인했다. Word2Vec는 시각화 결과 유사한 동작을 수행하는 연산자끼리 가까운 특징 공간으로 임베딩되는 것을 확인했다. [표 4]는 각 특징벡터의 크기별 가까운 니모닉을 나타낸 것이며, [그림 11]은 t-SNE로 임베딩된 니모닉을 시각화한 결과이다. 나머지 실험 결과는 6장 부록에 정리하였다.

[표 4] Word2Vec(Skip-gram)의 각 니모닉별 가장 가까운 위치에 사상된 니모닉

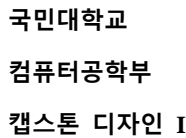
Mnemonic	vec 8	vec 16	vec 32	vec 64	vec 128
<b>mov</b>	lea	lea	lea	lea	lea
	setns	xor	push	push	push
	xor	cmovnz	xor	inc	inc
	test	push	xchg	xchg	sar
	cmovnz	cmovb	cmovnz	setnz	xor
	cmovz	shlx	cmovge	cmovge	setnz
	push	xchg	cmovz	sar	cmovge
	sets	cmovz	inc	xor	imul
	setz	add	sar	setnl	xchg
	cmovs	cmp	movsx	cwde	cmovg

<sup>3</sup> <https://radimrehurek.com/gensim/>

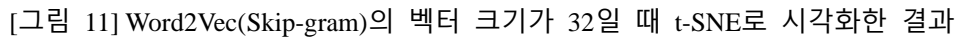


 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24

<b>jmp</b>	jge jl jnz jnb jbe lea jb jg mov ja	jl jb jge jnz jg ja retn jle jz jnb	retn jl jge jg jb jle jnz jz jns jbe	retn jg lea jle jnz jl jge js ja jz	retn jl jle jns jnz jge js jb jbe jz
<b>add</b>	pextrb dec prefetcht1 phaddsw cmovz cmp stmxcsr psrldq movd movq	sub shl prefetcht0 mov movups dec shlx adc inc cdq	sub shl prefetcht0 prefetchnta mov movaps sar pshufd cdq movups	sub shl imul sar mov lea neg subpd pshufd cmp	sub shl movaps mov imul lea jg movups sar neg
<b>pop</b>	leave xor fninit vzeroupper fldl2e cmpsd retn sfence fcmovb fcmovnb	retn leave vzeroupper xor sfence pushf popf cmpltd cmpnlesd cvtpd2pi	retn vzeroupper xor emms setz sfence leave setnl mov fninit	retn setz emms mov vzeroupper leave setns setnl setnz push	retn vzeroupper leave mov setz setnl setns jmp setnz push
<b>push</b>	jns lea setns js mov fnclex test stosd cmovs crc32	lea mov test fnclex jmp setns js fldl fldz cmovns	mov lea cmovs cmovnz cmovge cmovns test cmovl fnclex movsx	lea mov stosd setnz setz cwde test fstp cmovnz cmovz	mov lea test stosd setz setnz fstp cmovnz fldz inc



## 2020-MAY-24

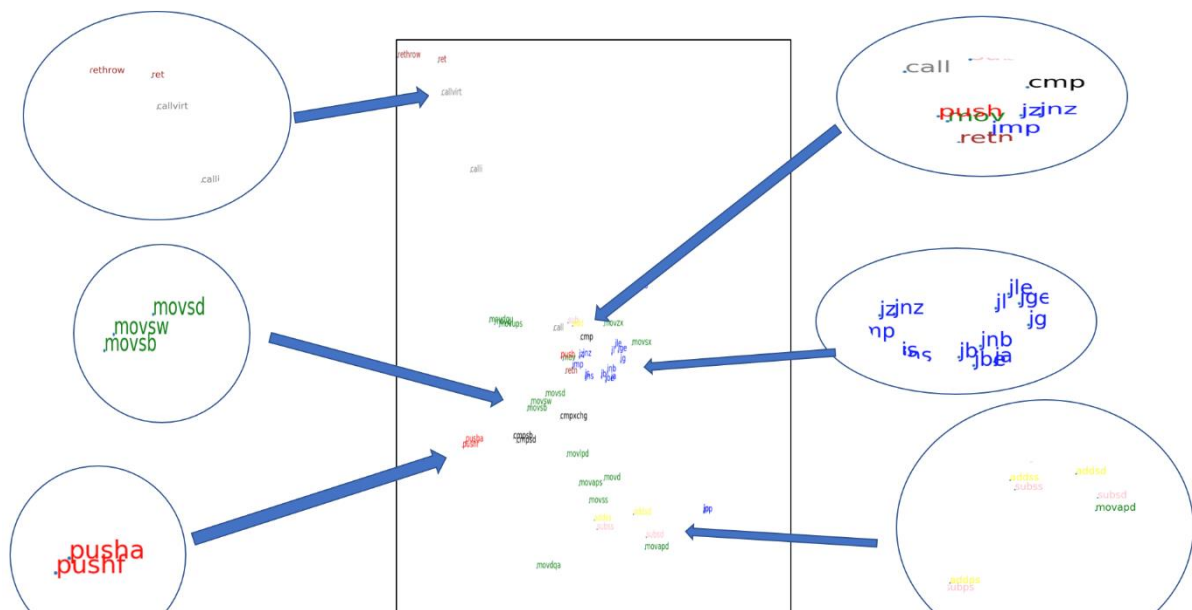


탐색할 하이퍼 파라미터는 특징벡터의 차원과 윈도우 크기이다. 특징벡터의 차원이 32, 64, 128, 256인 경우와 윈도우 크기가 2, 3, 4인 경우를 탐색한다. 임베딩 알고리즘 두 가지에 대해 실험을 진행하므로 총 24가지 경우에 대한 실험을 수행하였다. 나머지 하이퍼 파라미터는 최소 단

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24

어 수 1,000, 에포크 10, 학습률 0.0002로 지정했다.

실험 결과는 [그림 12]와 같다. Word2Vec(CBOW) 임베딩 알고리즘을 사용하여, 윈도우 크기는 2, 특징 벡터의 차원을 64로 하여 학습을 수행했을 때, 그림과 같이 유사한 동작을 수행하는 명령어끼리 가까운 위치로 임베딩 되었다.



[그림 12] Word2Vec(CBOW) 학습 결과

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24

## 2.2.4 딥러닝 기반 악성행위 추출 도구

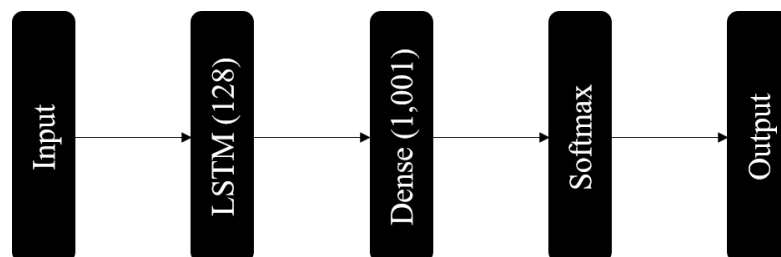
우리는 악성행위 추출을 위해 두 가지 방법을 사용하여 실험을 수행했다. 첫 번째는 [그림 3], [그림 4]와 같이 하나의 주변 명령어로부터 하나의 명령어를 예측해 예측한 명령어와 다른 경우 해당 명령어가 이상 행위를 수행한다고 추정하는 방식이다. 두 번째는 함수 단위로 RNN 기반 오토인코더(autoencoder)를 구현해 손실값이 특정 한계점보다 큰 경우 해당 함수가 이상 행위를 수행한다고 추정하는 방식이다. 실험에서 신경망은 Keras 라이브러리를 이용해 구현하였다.

### 가) 명령어별 이상탐지

주변 명령어로 하나의 명령어를 예측해 각 명령어의 이상 여부를 판정하는 방법은 다음과 같다.

1. 특징벡터를 임베딩 시킨다.
2. 특정 윈도우에 있는 명령어들을 입력 받아 다음 명령어, 혹은 중심 명령어를 예측한다.
3. 신경망이 예측한 명령어 중 상위 90% 이내에 포함되지 않으면 이상 명령어라 판정한다.

우리는 이전 명령어들로 다음 명령어를 예측하는 방식으로 실험을 수행했다. 하이퍼 파라미터는 다음과 같다. 단어 집합의 개수는 니모닉 1,000개와 알 수 없는 니모닉을 포함하여 1,001 개로 지정했으며, 윈도우 크기는 30으로 지정했다. 신경망은 [그림 13]과 같이 LSTM 한 층에 완전 연결층(fully-connected layer, dense layer)을 이은 신경망을 활용하였다.

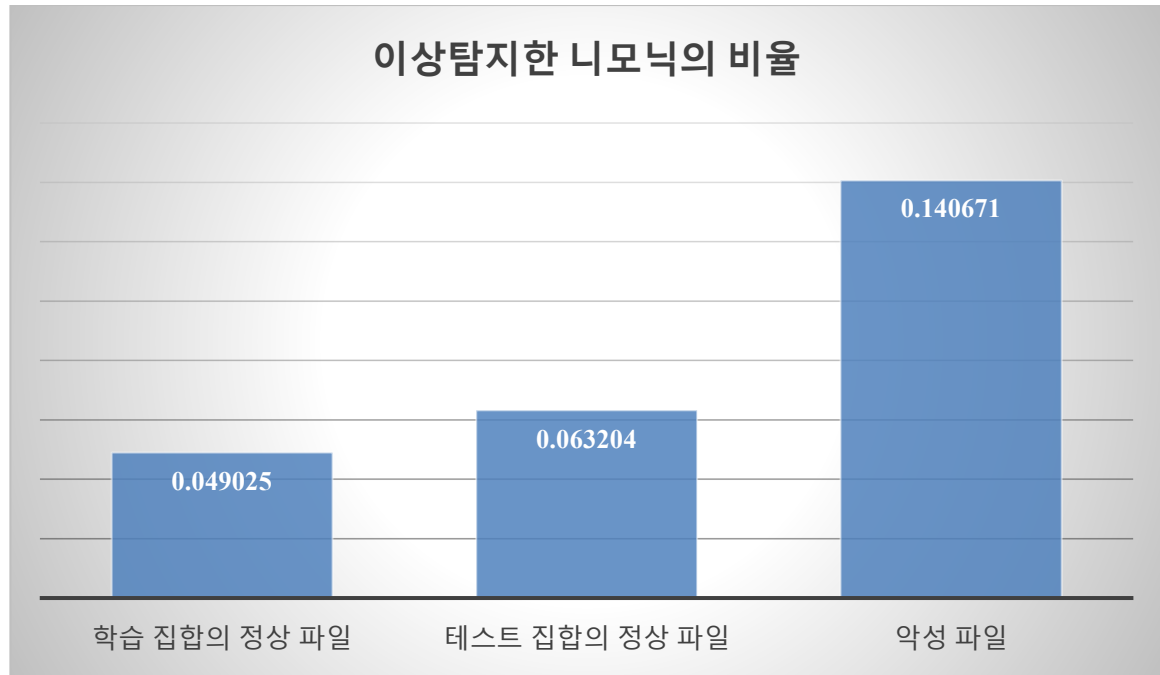


[그림 13] 명령어별 이상탐지를 위한 신경망의 구조

1,000개의 파일에 대해 학습을 수행하고, 학습 집합의 정상파일, 테스트 집합의 정상파일, 악성파일에 대해 이상탐지를 수행한 결과가 [그림 14]와 같다. 그림에서 악성파일에 대한 이상탐지 비율은 정상파일에 대한 이상탐지 비율의 두 배 이상임을 볼 수 있다. 그러나, 이 방법은 니모닉을 하나씩 예측하기 때문에 학습을 위한 시간이 많이 소요되고, 고사양의 컴퓨팅 환경을 필요로 하므로 실질적인 활용이 어렵다. 우리는 이를 개선하기 위해 여러 니모닉을 한 번에 추출하는 방

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24

식인 나)의 방법론을 적용할 예정이다.



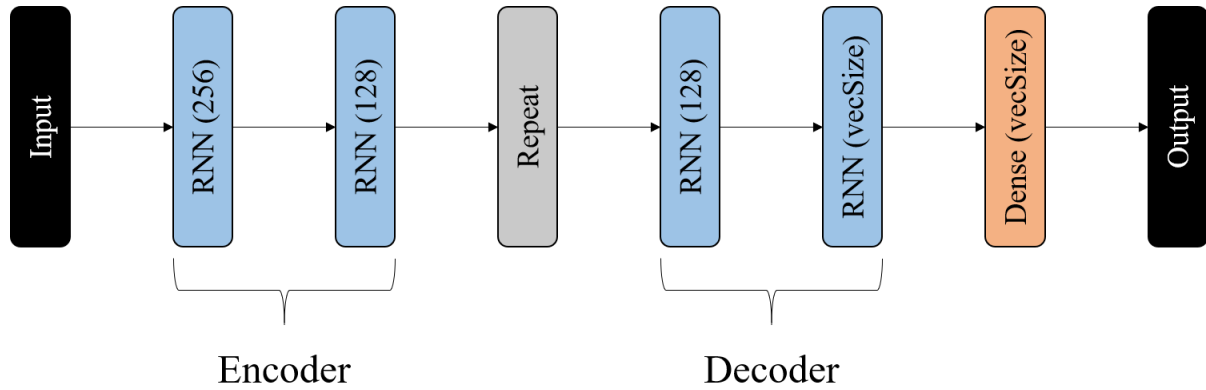
[그림 14] 테스트 집합별 이상탐지한 니모닉의 비율

### 나) 오토인코더 기반 이상탐지

오토인코더를 이용해 각 함수별 이상탐지를 수행하는 방법은 다음과 같다. 아래의 방법에서 우리는 한계치(threshold)을 0.1로 지정했다.

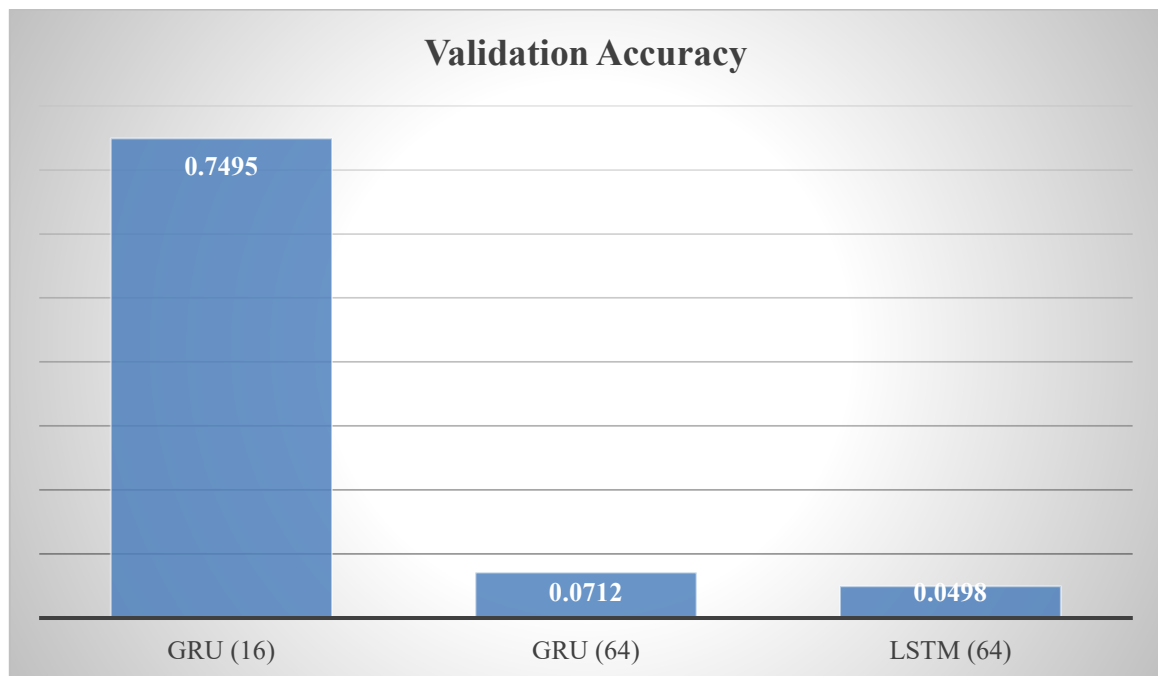
1. 특징벡터를 임베딩 시킨다.
2. 각 함수별 명령어를 입력 받아 해당 함수의 명령어를 예측한다.
3. 신경망이 예측한 명령어와 원본 명령어의 MAE(mean absolute error)를 계산하여, MAE가 특정 한계점보다 크면 이상행위를 수행하는 함수라 판정한다.

실험한 오토인코더의 구조는 [그림 15]와 같다. 그림에서 RNN으로는 GRU와 LSTM을 사용했으며 임베딩 벡터의 크기(vecSize)는 16, 64로 지정해 실험했다. 학습 데이터는 정상파일 10,000개를 사용했다.



[그림 15] 오토인코더 기반 이상탐지를 위한 신경망의 구조

신경망 구조별 학습 결과가 [그림 16]과 같다. 그림에서 x축은 사용한 RNN 구조와 특징벡터의 차원을 나타낸다. 학습 결과 GRU (16)은 테스트 집합의 손실값이 큰 반면, GRU (64)와 LSTM (64)는 낮다. 따라서 실험 결과는 2.2.3 목의 결과와 같이 특징벡터의 차원은 최소 32차원 이상이어야 함을 시사한다.

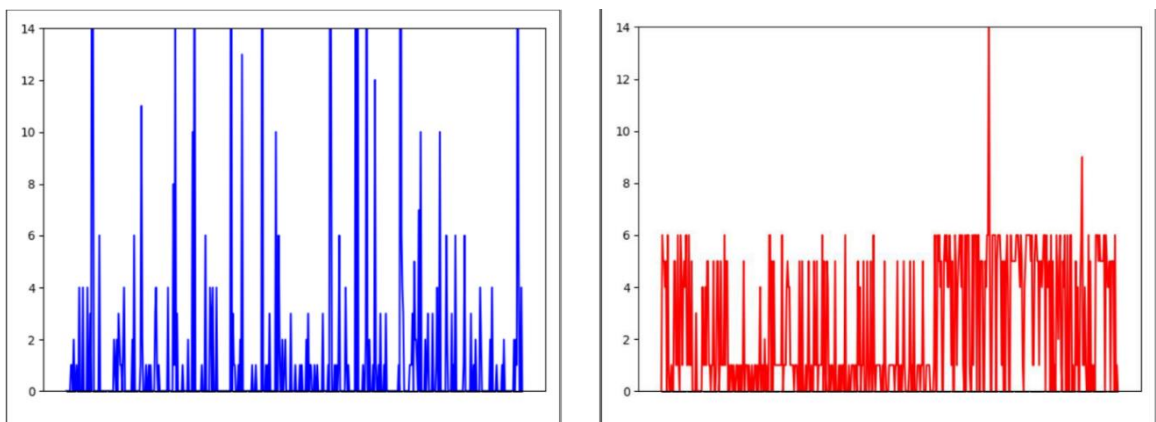


[그림 16] 신경망의 구조별 테스트 집합에 대한 손실값

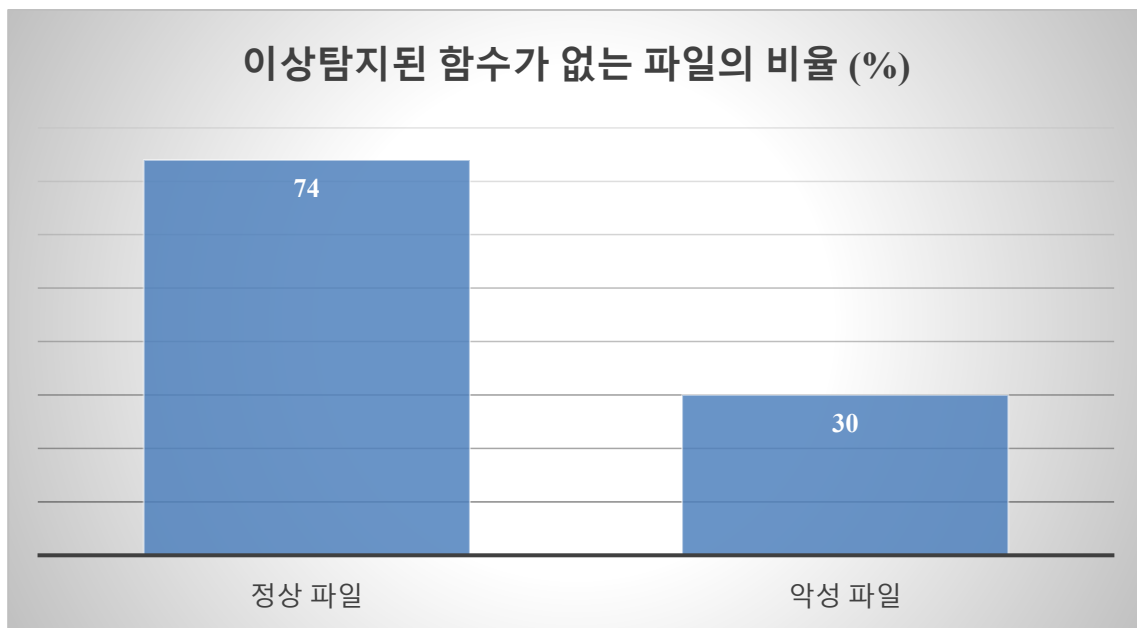
신경망이 학습을 제대로 수행했는지 확인하기 위해 정상파일 500개와 악성파일 500개에 대해 이상 탐지된 함수의 개수를 확인하였다. [그림 17]은 각 파일별 이상 탐지된 함수의 개수를 도식

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24

화한 것이다. 그림에서 정상파일은 이상탐지가 되지 않거나, 10개 이상의 많은 함수에 대해 이상탐지가 되는 반면, 악성파일은 대부분 6개의 함수에 대해 이상탐지가 되는 것을 볼 수 있다. 특정 정상파일의 함수들에 대해 이상탐지가 되는 이유는 학습에 충분히 많은 데이터를 사용하지 않았기 때문으로 추정된다. 한편, 모든 함수에 대해 이상탐지가 되지 않은 파일의 비율은 [그림 18]과 같다. 그림에서 정상파일은 전체 500개의 파일 중 약 74%가 이상탐지된 함수가 없는 반면, 악성파일은 약 30%가 이상탐지된 함수가 없었다.



[그림 17] 정상/악성파일의 이상탐지된 함수의 개수 (x축: 파일, y축: 이상탐지된 함수의 개수)  
(왼쪽: 정상파일, 오른쪽: 악성파일)



[그림 18] 이상탐지된 함수가 없는 파일의 비율

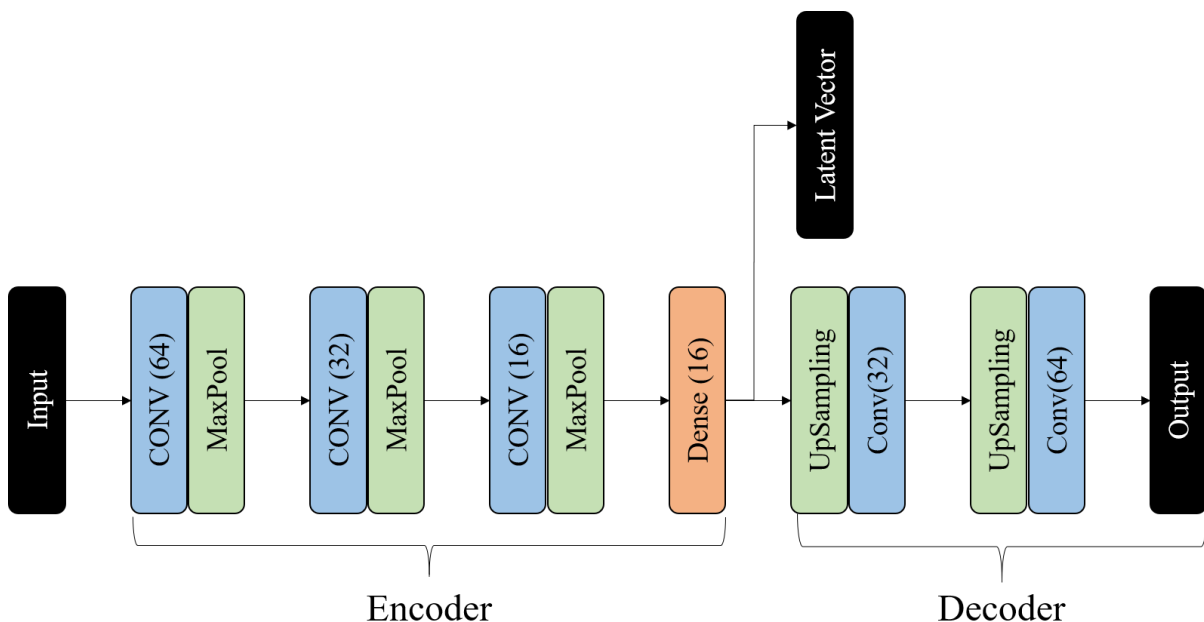
 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24

### 다) 압축벡터 기반 이상탐지

제안서의 방법은 실제 데이터에 대해 만족할 만한 성능을 보이지 못했다. 우리는 이를 보완하기 위하여 함수의 압축벡터(latent vector)를 추출하는 신경망을 구성하고, Elasticsearch를 기반으로 입력 함수와 유사한 함수를 찾아 함수의 악성 행위 가능성을 점수화하는 방법을 고안했다. 자세한 분석 방법은 다음과 같다.

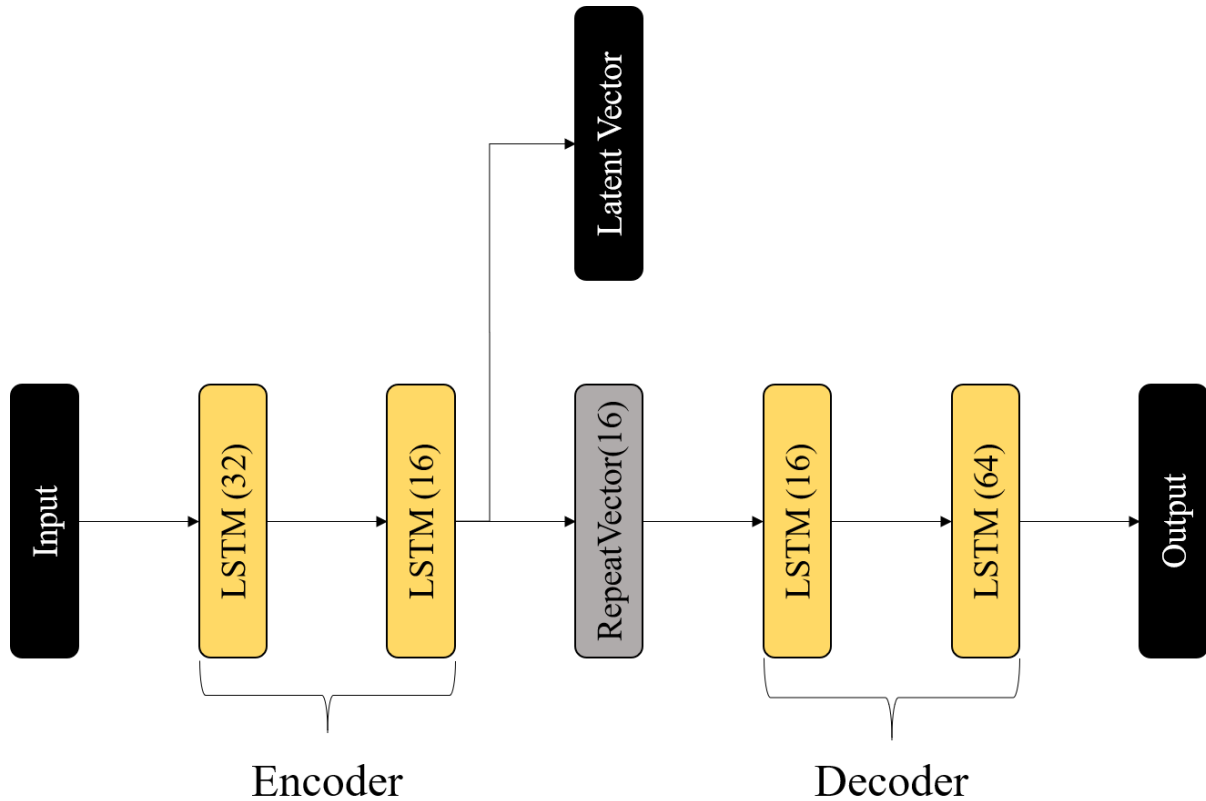
1. 신경망을 이용해 압축벡터를 구한다.
2. Elasticsearch로 데이터베이스에서 압축벡터와 코사인 유사도가 높은 함수를 찾는다.
3. 찾은 함수 중 악성 행위를 수행하는 함수의 비율로 입력 함수의 점수를 책정한다.

신경망은 [그림 19], [그림 20]과 같이 CNN 기반 오토인코더와 LSTM 기반 오토인코더를 구현하였으며 인코딩 결과를 압축벡터로 사용하였다. CNN 기반 오토인코더에서 컨볼루션층의 커널의 크기는 3, 풀링층의 커널의 크기는 2로 지정했으며, 업샘플링 크기도 풀링층의 커널의 크기와 동일하게 2로 지정했다. 두 신경망 모두 압축벡터와 출력을 계산하는 층에서 활성화함수는 sigmoid를 사용했으며, 이외의 층에서는 ReLU를 사용했다.



[그림 19] CNN 기반 오토인코더 구조

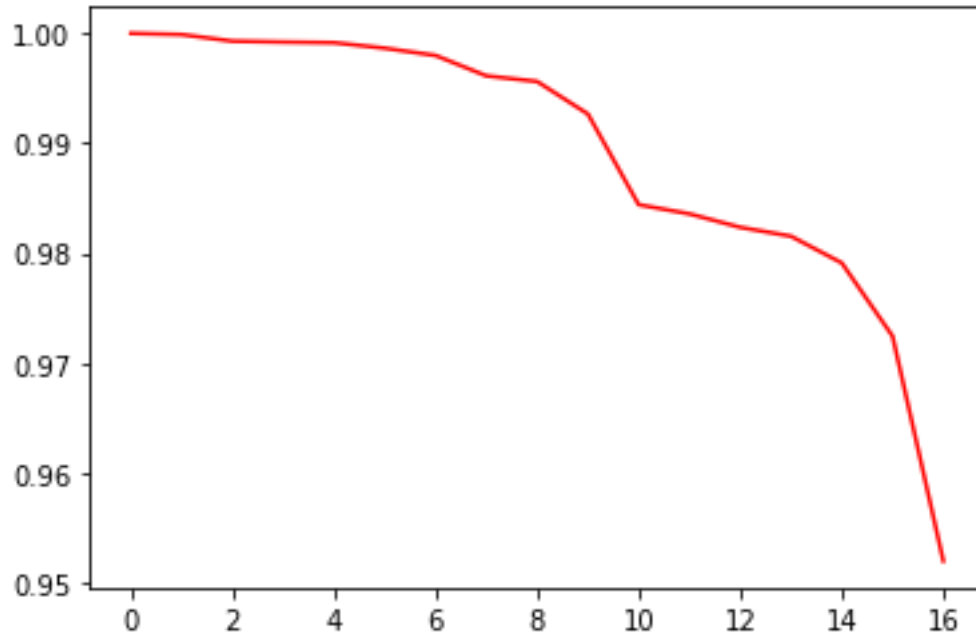




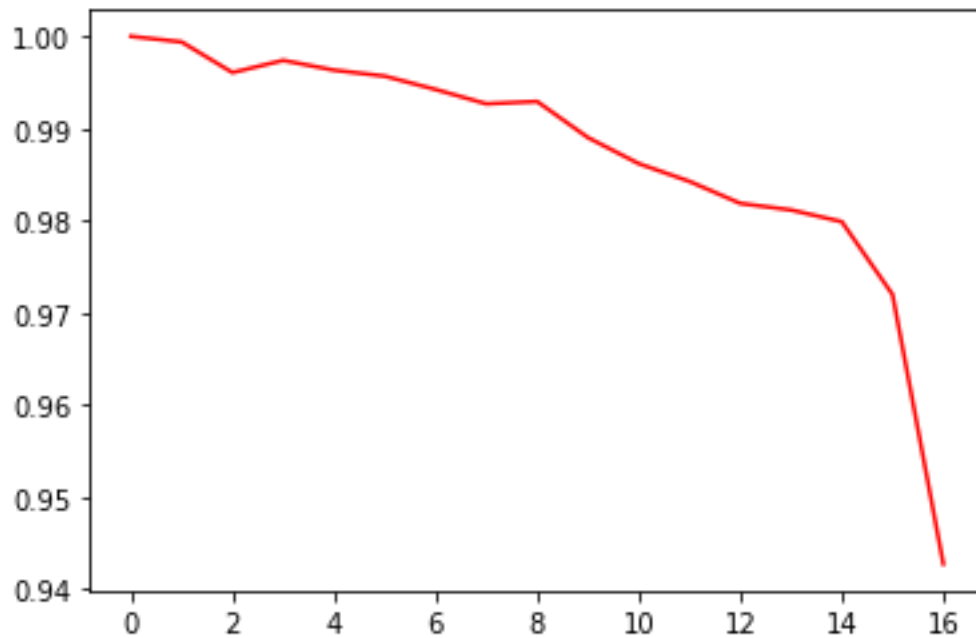
[그림 20] LSTM 기반 오토인코더의 구조

손실함수는 binary cross entropy, 최적화함수는 Adam으로(디폴트 파라미터) 하여, CNN은 16에포크, LSTM은 8에포크로 5,147,572개의 함수를 학습시킨 결과 마지막 손실값은 각각 0.6652, 0.6667 이었다.

제안한 방법의 적합성을 검증하기 위해 압축벡터의 코사인 유사도가 높은 것이 실제 두 함수가 유사함을 의미하는지 확인하였다. 함수의 유사성을 검증하기 위해 편집거리(edit-distance) 알고리즘을 사용하였다. [그림 21], [그림 22]는 편집거리별 압축벡터의 코사인 유사도의 평균을 나타낸 것이다. 그림에서 x축은 편집거리, y축은 코사인 유사도의 평균을 의미한다. 그림에서 편집거리가 증가하면 코사인 유사도가 감소한다. 따라서 제안한 방법은 함수의 유사도를 측정하기에 적합하다.



[그림 21] CNN 기반 인코딩 결과 코사인 유사도와 편집거리

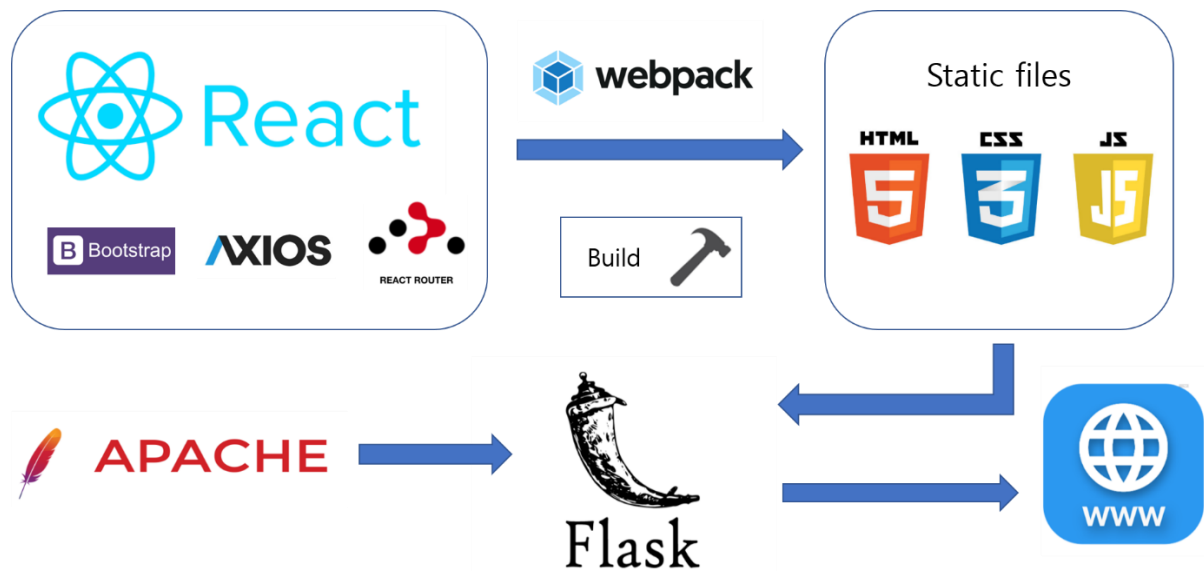


[그림 22] LSTM 기반 인코딩 결과 코사인 유사도와 편집거리

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	2차 중간보고서		
	프로젝트 명	asi(a security insight)	
	팀 명	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24

### 2.2.5 웹 서비스

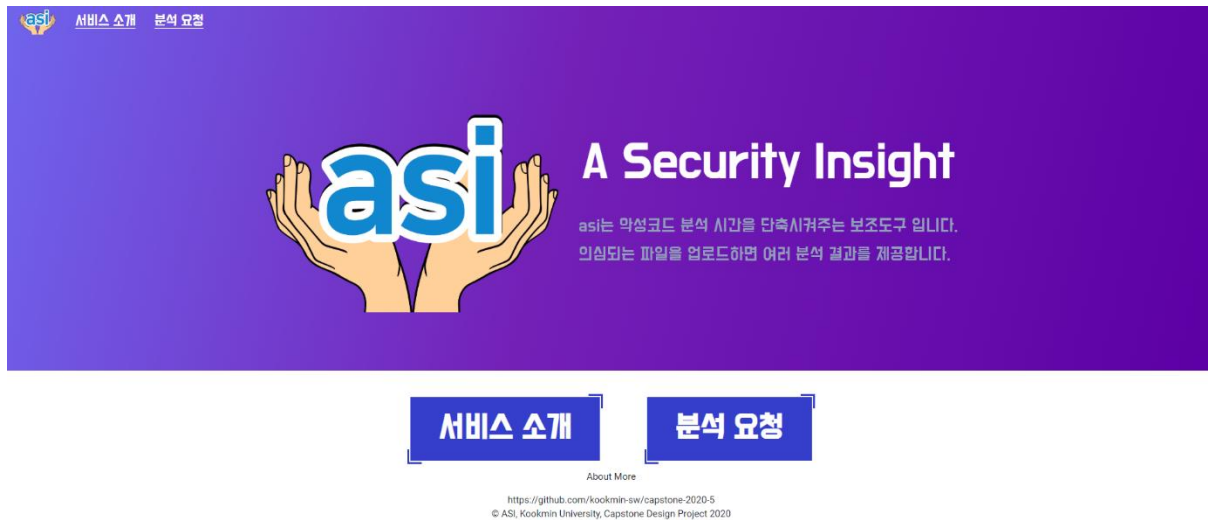
사용자 친화적인 인터페이스 구축을 위해 웹 서비스를 구축하였다. 웹 구축을 위한 개발 환경은 [그림 23]과 같다. 프론트엔드(frontend)는 React 라이브러리를 기반으로 만들었으며 Axios 라이브러리를 이용해 Flask와 React 간의 파일 송수신을 구현했다. 서버는 Apache http로 구현하였다.



[그림 23] 웹 서비스 구축을 위한 라이브러리 구조

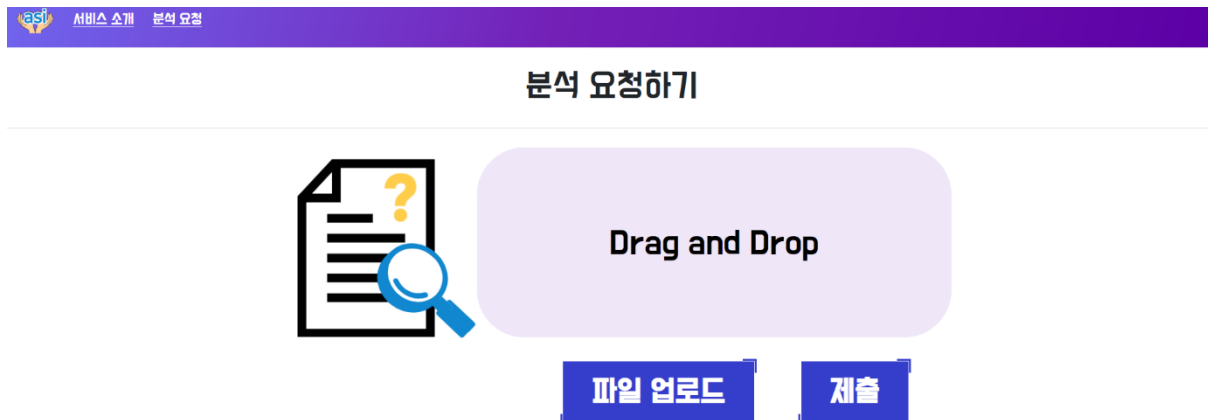
웹 서비스를 구현한 결과가 다음과 같다. [그림 24]는 메인 페이지로 사용자는 본 프로젝트에 대한 소개를 확인할 수 있으며, 분석을 수행할 수 있다.

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24




[그림 24] 초기 화면

분석 요청 버튼을 누르면 [그림 25]와 같이 파일을 업로드 할 수 있는 화면이 나타나며, 사용자는 drag-and-drop 방식을 사용하거나, 파일 업로드 버튼을 눌러 파일 탐색기에서 파일을 찾아 업로드 할 수도 있다.



[그림 25] 분석 파일 업로드 화면

[그림 26]은 분석 결과 화면으로 사용자는 분석된 파일의 목록을 확인할 수 있다. 현재 각 파일 별 분석 결과를 확인하는 화면은 구현되었지만, 메타 데이터 출력과 전체 결과 다운로드 기능







 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24

은 구현되지 않았다.




## Upload / File List

[Download ZIP](#)

파일명	결과	정보
 00d7ab9a8e5c9a84cfa19ad9e583e6f.exe	<a href="#">결과 보기</a>	@meta
 00d9dada816ad75caf879e09de2a521.exe	<a href="#">결과 보기</a>	@meta
 00d8914ba4c475e568a292afbd7b35d1.exe	<a href="#">결과 보기</a>	@meta
 00dde3d759f73ba093da9a375d725f47.exe	<a href="#">결과 보기</a>	@meta
 0000f5c78ed2442813ceef6afa1436c3.exe	<a href="#">결과 보기</a>	@meta
 00d2c06a552f782c1f16acf77db765a5.exe	<a href="#">결과 보기</a>	@meta

List of users


[그림 26] 분석 결과 목록 화면

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24

## 3 수정된 연구내용 및 추진 방향

### 3.1 수정사항

제안서의 방안은 실험 결과 악성 행위 수행 여부를 제대로 판정할 수 없었고, 전문가의 분석 결과와도 차이가 있는 것을 확인했다. 우리는 2.2.4 목의 다)항에서 제시한 것과 같이, 오토인코더를 학습시키고, 인코딩 결과인 압축벡터의 코사인 유사도를 기반으로 데이터베이스의 함수 중 입력 함수와 유사한 함수를 추출한 뒤, 해당 함수들 중 악성 행위를 수행하는 함수의 비중을 계산해 입력 함수의 악성 행위 수행 가능성을 점수화하는 방안을 개발하였다.

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24

## 4 향후 추진계획

### 4.1 향후 계획의 세부 내용

#### 4.1.1 딥러닝 기반 악성행위 추출 도구

현재 단일 모델에 대해 학습을 수행하였고, 하이퍼 파라미터는 디폴트 파라미터를 활용하였다. 이후 다양한 모델과 하이퍼 파라미터에 대한 실험을 수행하여 최적의 신경망을 찾을 예정이다. 또한, 함수의 유사성을 편집거리만으로 검증하였는데, 추후 다른 알고리즘을 적용하여 함수의 유사성을 검증할 예정이다.

#### 4.1.2 웹 서비스

아직 구현되지 않은 파일의 메타 데이터를 출력하는 기능과 전체 분석 결과를 다운로드 하는 기능을 구현할 예정이며, 사용자 매뉴얼을 추가할 예정이다. 분석 결과 화면에서 각 필드 (mnemonic, strings, import, export)별로 세부 데이터를 출력하는 기능과 Elasticsearch 결과를 시각화하는 기능을 추가할 예정이며, mnemonic의 계열별로 다른 색상의 글씨로 출력하도록 수정할 것이다.

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24

## 5 고충 및 건의사항

없음.



 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24

## 6 부록

[표 5] Word2Vec(CBOW)의 각 니모닉별 가장 가까운 위치에 사상된 니모닉

Mnemonic	vec 8	vec 16	vec 32	vec 64	vec 128
<b>mov</b>	jmp	jmp	jnz	jmp	jmp
	lea	jle	jmp	jle	jz
	jz	jnz	jle	jnz	jnz
	jnz	xor	jz	jz	jle
	jle	jbe	jg	jg	jge
	jbe	jz	xor	xor	lea
	call	jge	jge	jge	jl
	push	jl	jbe	lea	xor
	jge	jg	jl	jbe	jg
	jl	ldarg.1	cmpunordps	jl	push
<b>jmp</b>	mov	mov	jnz	mov	mov
	jnz	jnz	mov	lea	lea
	jl	ret	jz	jnz	jnz
	jz	jz	jg	jz	jz
	jbe	lea	jle	call	call
	jge	jl	lea	setz	jge
	jg	ldarg.1	jge	jge	jl
	jle	box	setnz	jle	push
	lea	jle	setz	jl	jle
	call	jge	call	jg	jg
<b>add</b>	jbe	ldloc.2	ldc.i4.1	imul	jb
	ja	imul	imul	ja	ja
	sub	sub	ldc.i4.3	jb	jg
	jnb	adc	ldc.i4.2	jnb	jnb
	jg	stloc.3	jg	jg	imul
	jle	ldloc.3	ja	jno	sub
	jge	ldloc.0	stelem.i4	adc	jl
	jz	ja	ldelem.i4	sub	jbe
	jnz	stloc.1	sub	jbe	adc
	jb	sar	bge.s	jl	shl
<b>pop</b>	leave	leave	retn	leave	retn
	retn	retn	leave	retn	leave
	cgt.un	callvirt	ldsflld	vfmsub231ss	mov
	pushf	castclass	callvirt	divpd	xor
	stosd	brfalse.s	ldelem.ref	vfmsub213sd	leave.s

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24

	setns vzeroupper fsincos ldnull fclex	isinst ldelem.ref blt.s stloc.2 btrue.s	xor pushf stloc.s vzeroupper mov	vpminsd vpckusdw movdq2q pextrb pmovzxd	jmp vcmpgtsd cmpsd vdivss call
<b>push</b>	lea call mov jmp test throw ldtoken vfmsub231sd jl jge	lea call jz jnz mov jle jmp stloc.0 ldarg jge	call lea jz jnz js jmp box jle dup ldc.i4.0	lea call jz jnz jmp jle mov js setnz test	lea call jz jnz jmp mov jle test js jl

[표 6] FastText(Skip-Gram)의 각 니모닉별 가장 가까운 위치에 사상된 니모닉

Mnemonic	vec 8	vec 16	vec 32	vec 64	vec 128
<b>mov</b>	setz	lea	lea	lea	lea
	cmovnz	push	xor	push	push
	cmovz	add	push	xor	add
	setnz	vmoovsd	shlx	add	xor
	setns	shlx	add	xchg	inc
	xor	cmp	shl	inc	cmp
	setnl	xor	sar	setnz	cmovz
	sets	inc	inc	vmoovsd	movzxd
	setl	shl	setnl	cmp	moovsd
	test	jno	cmovz	sar	setz
<b>jmp</b>	jl	jl	retn	retn	retn
	jbe	jge	jl	jle	jle
	jb	jb	jle	jg	jl
	jnz	jnb	jge	jl	jnz
	jge	ja	jb	jge	jge
	jnb	jbe	jg	jns	jg
	ja	jg	jnb	jnz	jns
	jle	jle	jbe	lea	js
	jg	retn	jnz	js	jz
	fsincos	jnz	js	jb	jb
<b>add</b>	dec	sub	sub	sub	sub
	cmpneqsd	prefetcht0	prefetcht0	shl	mov

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24

	cmovbe setnbe stmxcscr ucomisd btr cvtss2si maxpd cmp	cmpltpd prefetcht1 mov movups adc cmpnlepd movddup prefetchnta	prefetcht1 shl prefetch sar unpckhpd imul mov prefetchnta	mov prefetcht1 prefetcht0 lea cmp imul prefetch pmulldq	imul lea cmp shl movaps xorps addpd mulpd
<b>pop</b>	leave popf pushf jecxz sti stc vroundsd vzeroupper cld loopne	ret retf leave vzeroupper cmpltd popfw cmplsd pushf xor les	ret popfw cmpsw xor roundsd cmpsd popf vzeroupper vroundsd vcmpgtsd	ret leave push vroundsd popf cld setz roundsd vpbroadcastw popfw	ret push leave mov call lea cmplsd vzeroupper jmp setz
<b>push</b>	jns js lea fnclex call stosd test cmovs setns jz	lea mov setns fnclex haddpd fsincos cmovns cmovnz cmovle cmovz	lea mov test fnclex setns cmovs setnz fldl xor setz	lea mov test setnz setz cmovnz stosd xor cmovns ficom	lea mov setnz setz test stosd cmovnz jz xor setns

[표 7] FastText(CBOW)의 각 니모닉별 가장 가까운 위치에 사상된 니모닉

Mnemonic	vec 8	vec 16	vec 32	vec 64	vec 128
<b>mov</b>	jmp	jmp	jle	jle	jmp
	jg	jle	jnz	jnz	jnz
	jl	jbe	jmp	jz	jle
	cmovnz	jnz	jge	jmp	jz
	jnz	jge	jz	jge	jge
	jge	push	jl	jg	jl
	jb	jz	jg	xor	jg
	setz	jnb	jbe	jbe	jnb
	jbe	jl	jnb	jnb	push
	ldloc.3	lea	jb	jl	jbe



<b>jmp</b>	mov jg jl ldloc.3 cmovnz jnz jge jb ldloc.2 jbe	mov jl jb jnz jge jbe fcmovnb lea fcmovnb jg	mov jge lea jl jnz jg jle jb pushfw ja	mov lea jge jnz movss setp push jg jle jnb	mov lea movss jge jnz jl push movaps test jz
<b>add</b>	ldind.ref ldelem ldelema stmxcscr movzx ldelem.r8 ldelem.r4 endfinally stloc starg.s	movddup movshdup movsldup inc imul movntq movd movnti movsx movntdq	jno imul adc psignw movnti cvttps2dq fcmovb cvtps2dq paddusw pavgw	jno shlx jg imul adc movnti fiadd cmovb fimul movntdq	imul jg adc fimul fiadd addsd jno jl sub ja
<b>pop</b>	leave newobj retn ldnull ldvirtftn rethrow ldsflld ret leave.s iret	retn leave popfw popcnt brtrue popf fsave lds callvirt brtrue.s	retn leave popfw popcnt popf popa castclass leave.s isinst pushf	retn leave popfw leave.s popf movhlps popcnt br.s movlhps ldsflld	retn leave popfw leave.s popf popa popcnt xor br.s ldsfllda
<b>push</b>	lea call jl jmp mov ldelem.ref jle jge ldstr jnz	call lea ldtoken mov ldstr pushfw box calli jnz pusha	lea call pushfw calli jmp jnz jz pusha pmulhw mov	lea call pushfw pusha jnz jz jmp mov jle jge	lea call pushfw jmp jnz jz mov jl pusha test

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24

[표 8] Doc2Vec(distributed-memory)의 각 니모닉별 가장 가까운 위치에 사상된 니모닉

Mnemonic	vec 8	vec 16	vec 32	vec 64	vec 128
<b>mov</b>	lea	lea	push	push	push
	stosd	push	movss	lea	lea
	xor	xor	pushf	cpuid	vfmsub213sd
	push	movss	xchg	aesenc	vfmsub231ss
	stosw	cmpsb	vmovsd	movaps	vfmsub231sd
	movsd	xorps	movq	cmp	cmp
	cmpxchg	cmpsd	lea	xadd	movzx
	sar	sar	bswap	movss	vfnmadd213ss
	fldz	fst	paddsb	xchg	vpclmulqdq
	stosb	jecxz	cpuid	imul	vfmsub213ss
<b>jmp</b>	jno	retn	retn	retn	retn
	retn	vpmuldq	jns	jnz	jnz
	jnb	vfmsub213ps	js	js	js
	jb	vhaddps	jle	jle	jz
	jl	retf	jnz	jns	jle
	ja	ldloc.2	jg	jz	jns
	jbe	jl	ja	jl	jl
	ror	hlt	jnb	jb	jb
	jge	vpmullw	jb	jnb	jnb
	ud2	loope	pxor	jbe	jge
<b>add</b>	sub	vbroadcastsd	conv.i	ldelem.i2	add.ovf
	switch	vmovaps	mul	vaddss	blt.un
	cmpltss	vmovddup	mul.ovf.un	vmovntdq	rem.un
	cbw	prefetcht0	vmovntdq	vminps	blt.un.s
	ldc.i4.8	prefetchnta	inc	add.ovf	ldelem.i2
	ldc.i4.4	pavgb	add.ovf	vcvtps2dq	sub.ovf
	and	vbroadcastss	ldind.u2	vmaxps	bge.un
	ldc.i4.1	vunpcklpd	pshufb	vmovlps	vpadd
	vldmxcsr	vmovq	vmovdqu	bne.un.s	ble.un
	ldc.i4.6	vmovhps	fadd	vpackuswb	div.un
<b>pop</b>	leave	leave	leave	leave	leave
	sfence	vsrld	jecxz	vhaddps	jmp
	endfinally	cmpltss	retn	jecxz	callvirt
	vzeroupper	cvtpd2pi	castclass	cvttps2pi	vpmuldq
	lodsd	vpslldq	retf	retf	jecxz
	pushf	vpmuldq	vzeroupper	loope	cvttps2pi
	jecxz	vfmsub213ss	cmpltss	andnpd	iret
	brtrue	vzeroupper	loop	vfmsub231ps	loop
	brfalse	vfnmadd132sd	jo	iret	retn
	ror	vcmpeqps	iret	vcmpgt_oqps	hlt

 <b>국민대학교</b> <b>컴퓨터공학부</b> <b>캡스톤 디자인 I</b>	<b>2차 중간보고서</b>		
	<b>프로젝트 명</b>	asi(a security insight)	
	<b>팀 명</b>	assist(a security safety important special team)	
	Confidential Restricted	Version 1.1	2020-MAY-24

<b>push</b>	lea	lea	mov	mov	mov
	mov	mov	lea	vfmsub231ss	vfmsub231sd
	call	pmulhw	paddsb	lea	vfmsub213sd
	fldz	vcmpnle_uqps	movss	vfmsub231sd	lea
	setnle	movsx	test	vsubss	vfmadd231sd
	stosd	vcmpole_oqps	paddusb	vfmadd231sd	vcmpgt_oqps
	ldtoken	psubusw	vaesenc	vfmsub213sd	vfmsub231ss
	fucom	movq	subps	maxps	cmpnleps
	fst	psubw	movdqa	packssdw	maxps
	fstp	vfmadd231sd	vpxor	vmaxss	minps

[표 9] Doc2Vec(DBOW)의 각 니모닉별 가장 가까운 위치에 사상된 니모닉

Mnemonic	vec 8	vec 16	vec 32	vec 64	vec 128
<b>mov</b>	lea	lea	push	push	push
	stosd	push	movss	lea	lea
	xor	xor	pushf	cpuid	vfmsub213sd
	push	movss	xchg	aesenc	vfmsub231ss
	stosw	cmpsb	vmovsd	movaps	vfmsub231sd
	movsd	xorps	movq	cmp	cmp
	cmpxchg	cmpsd	lea	xadd	movzx
	sar	sar	bswap	movss	vfmadd213ss
	fldz	fst	paddsb	xchg	vpclmulqdq
	stosb	jecxz	cpuid	imul	vfmsub213ss
<b>jmp</b>	jno	retn	retn	retn	retn
	retn	vpmuldq	jns	jnz	jnz
	jnb	vfmsub213ps	js	js	js
	jb	vhaddps	jle	jle	jz
	jl	retf	jnz	jns	jle
	ja	ldloc.2	jg	jz	jns
	jbe	jl	ja	jl	jl
	ror	hlt	jnb	jb	jb
	jge	vpmullw	jb	jnb	jnb
	ud2	loope	pxor	jbe	jge
<b>add</b>	sub	vbroadcastsd	conv.i	ldelem.i2	add.ovf
	switch	vmovaps	mul	vaddss	blt.un
	cmpltss	vmovddup	mul.ovf.un	vmovntdq	rem.un
	cbw	prefetcht0	vmovntdq	vminps	blt.un.s
	ldc.i4.8	prefetchnta	inc	add.ovf	ldelem.i2
	ldc.i4.4	pavgb	add.ovf	vcvtps2dq	sub.ovf
	and	vbroadcastss	ldind.u2	vmaxps	bge.un
	ldc.i4.1	vunpcklpd	pshufb	vmovlps	vpadd
	vldmxcsr	vmovq	vmovdqu	bne.un.s	ble.un



	ldc.i4.6	vmovhps	fadd	vpackuswb	div.un
pop	leave	leave	leave	leave	leave
	sfence	vpsrld	jecxz	vhaddps	jmp
	endfinally	cmpltss	retn	jecxz	callvirt
	vzeroupper	cvtpd2pi	castclass	cvttps2pi	vpmuldq
	lodsd	vpslldq	retf	retf	jecxz
	pushf	vpmuldq	vzeroupper	loope	cvttps2pi
	jecxz	vfmsub213ss	cmpltd	andnpd	iret
	brtrue	vzeroupper	loop	vfmsub213ps	loop
	brfalse	vfmadd132sd	jo	iret	retn
	ror	vcmpeqps	iret	vcmpgt_oqps	hlt
push	lea	lea	mov	mov	mov
	mov	mov	lea	vfmsub213ss	vfmsub213sd
	call	pmulhw	paddsb	lea	vfmsub213sd
	fldz	vcmpnle_uqps	movss	vfmsub213sd	lea
	setnle	movsx	test	vsubss	vfmadd213sd
	stosd	vcmpole_oqps	paddusb	vfmadd213sd	vcmpgt_oqps
	ldtoken	pbusw	vaesenc	vfmsub213sd	vfmsub213ss
	fucom	movq	subps	maxps	cmpnleps
	fst	psubw	movdqa	packssdw	maxps
	fstp	vfmadd213sd	vp xor	vmaxss	minps