단어의 의미적 유사도를 응용한 암호 생성기

컴퓨터공학과 15학번 최현호, 18학번 김재웅

팀명: DLN(Deep-Learning-Needed) / 지도교수: 이영구 교수님

요약

문장 간의 유사도를 측정하는 NLP 방식을 활용하여, 단어의 정의문 및 단어 용례의 유사도 검증을 수행한, 유사도가 떨어지는 단어의 조합으로 암호를 생성하여 사용자의 숙지 용이성 및 암호의 안전성을 동시에 확보한다. 암호의 안전성은 기존의 난수화 암호와 동일한 길이의 암호를 생성하여 검증한다. 이를 통해 새로운 형태의 무작위 암호 생성 방법을 제공한다.

1. 서론

1.1. 연구배경

현재, 컴퓨터의 연산 능력이나 각종 처리 기술이 발전함에 따라 강력한 암호화 기술이나 블록체인 기술을 통한 부인 방지 및 변조 방지 기술 등이 꾸준히 개발되고 있다. 물론, 지금은 지문 인식이나 OTP(One-Time-Password) 등의 편리한 인증 수단이 출시되고 있지만, 아직까지 재래식 패스워드를 사용하여 계정 암호를 관리하는 집단이 적지 않다. 2020년 FBI에서 게재한 '비밀번호를 통한 디지털 방호 구축' 기사에서, "DirectorMonthLearnTruck"와 같은 의미적 유사도가 낮은 단어의 조합으로 Passphrase(이하 비밀구절)를 구축하는 것이 짧고 복잡한 Password(이하 암호)를 구축하는 것보다 암기의 측면과 보안의 측면에서도 안전하다고 밝혔다.[1] 또한 여러 문자와 숫자, 특수문자를 포함한 복잡한 암호보다는 15자 이상의 단어로 이뤄진 긴 문장형으로 된 비밀번호가 보안성이 높다는 분석도 나왔다.[2] Google Chrome과 같은 사이트 암호 저장을 지원하는 브라우저에서는 난수화한 암호를 생성하는 기능 역시 지원하는데, 정작 해당 기능을 사용하는 Google 계정에 대해서는 해당 보호를 적용하기 어렵다는 단점이 있다. 따라서, 2~4글자의 의미적 유사도가 낮은 단어를 최소 2개 이상 생성하여, 암호를 생성하는 서비스에 대하여 구상하였다.

1.2. 연구목표

현재 서비스 중인 암호 생성 서비스는 사용자의 생성 명령 입력 시에 즉각적으로 암호를 출력하는 실시간 서비스이다. 하지만, 암호 생성기로 생성되는 암호는 비밀구절과 달리 난독화, 난수화 작업이 진행되었기 때문에 숙지 난이도가 높고, 가독성이 떨어진다. 따라서, 해당 서비스의 이

점을 최대한으로 살리면서, 생성되는 비밀구절의 보안성 및 숙지 용이성을 최대한으로 끌어내도록 구현한다.

모국어 발음과 실제 입력이 Qwerty 자판(이하 쿼티)를 기반으로 이루어지는 영어나 중국어, 일본어 등과 달리, 한국어는 키보드 입력 방식인 두벌식 표준이 쿼티에 대응되기 때문에, 단어를 쿼티 기반으로 변환했을 때 의미를 한 눈에 파악하기 어렵다는 점이 있다. 따라서, 한국어 자판 배열의 비밀구절을 생성하여 암호 생성의 이점으로 활용한다.

생성되는 비밀구절 후보 단어는 일정 유사도 미만의 조건을 만족해야 한다. 또한, 데이터베이스에 등록된 단어는 발음에 따라 숫자 혹은 알파벳으로 치환할 수 있는 글자 (‘이’나 ‘투’라면 2, ‘티’나 ‘비’는 각각 ‘t’와 ‘b’)로 치환한다. 해당 과정을 통해 사전 공격(Dictionary Attack) 취약점을 보완한다. 위 과정을 거친 암호를 시스템의 보안 정책에 부합하는지 검증한다. 검증이 완료된 구절은 숙지를 위해 가공되지 않은 원본과 발음에 따라 치환한 구절을 제공하여 숙지 용이성을 끌어올린다.

2. 관련연구

2.1. 단어의 의미적 유사도 분석

NLP(Natural Language Processing, 이하 자연어처리)의 일환으로, 두 문장 속의 단어만을 벡터화하여 유사도를 비교하는 방법이 문장 간 유사도 분석 방법이다. 기존에 구축된 WordNet(프린스턴 영어에 대한 대규모 어휘 데이터베이스)이나 표준국어대사전에 등재된 단어와, 그 의미를 토대로 해당 단어에 대한 정의문과 용례를 각각 sent2vec 및 word2vec 모델을 통해 벡터화하여 단어 의미 유사도를 측정하는 모델에 관한 논문이 2018년 한국콘텐츠학회의 학술대회 논문집에 게재되었다.[3)]

또한, 위 논문의 연구 배경이 되는 기존 기술인, ETRI가 표준국어대사전을 기반으로 구축한 WiseWordNet은 현재 Open API 형태로 제공 중이다.

2.2. 무작위 암호 생성기

현재 계정 보안을 위하여 실제로 운용되고 있는 서비스로는, Windows의 도메인 환경에서 관리자 권한 임시 부여를 위해, 짧은 시간 내에 만료되는 암호를 로컬 시스템의 관리자 계정에 적용하는 LAPS(Local Administrator Password Solution)이 대표적이다. 다만, 랜덤 생성된 암호는 알파벳 대소문자와 특수기호, 숫자를 조합하였는데 유사한 모양의 텍스트로 인해 가독성이 매우 떨어져, 랜덤 생성된 암호를 성공적으로 전달하는 것 역시 쉽지 않다. 다만 암호 만료 기간 및 재생성할 암호를 주기적으로 갱신되는 그룹 정책에 탑재하여 관리한다는 점이 용이할 뿐이다.

또한, 앞서 언급한 Google Chrome의 '안전한 비밀번호 추천' 기능이 있다. 계정 가입 절차에 돌입했을 때, 암호 입력 상자에 진입하면, '안전한 비밀번호 추천' 팝업에서 LAPS와 유사한 형태의 난수화된 암호를 제공하고, 해당 암호를 Google 계정에 자동으로 저장하는 방식이다. 해당 계정을 통해 암호를 설정한 시스템의 암호 보안은 상대적으로 안전하겠지만, 암호가 저장된 Google 계정을 사용하지 않으면, 해당 기능을 통해 생성한 암호를 이용하는 사이트는 사실상 이용이 불가능하다는 단점이 있다. 또한, 해당 Google 계정에 대한 암호는 해당 기능을 통해 암호화할 수 없어, 상대적으로 취약한 암호를 사용하거나, 다른 인증 수단을 도입하는 방법밖에 없다.

2.3. 기존 연구 활용 방안

문장이나 문단, 혹은 문서 간의 유사도 분석에 주로 이용되는 유사도 검사는 단어 부문에서는 아직까지 의미적으로 유사도가 매우 높은 단어를 추천하는 시스템이나, 유사도를 기반으로 문장 감성 분석 모델로 활용하여 영화 리뷰 등을 분류하는 등, 높은 유사도 수치를 기반으로 한 서비스가 주로 구상되었으며, 낮은 유사도를 기반으로 한 서비스는 그다지 고안되지 않았다. 따라서, 벡터화 된 유사한 단어들을 한 범주로 묶어주는 유사도 분석 모델을 역이용한다면, 랜덤으로 생성된 단어 간의 유사도 검사가 가능하고, 검사 결과 유사도가 0에 가까운 단어를 이용한다면, 의미적으로 연상하기는 어려우나, 난수화 된 암호에 비해 외우기 쉬운 비밀구절을 생성할 수 있다.

뿐만 아니라, 앞서 언급한 ETRI의 WiseWordNet의 Open API를 본 연구를 통해 자체 구축할 예정인 모델과 비교하여, 딥러닝 모델의 구현 정도를 검증할 수 있다.

3. 프로젝트 내용

3.1. 시나리오

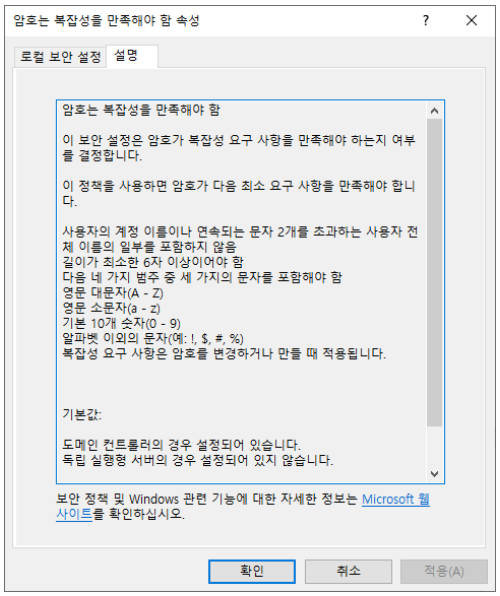
사용자 측면에서의 UI 및 사용 기능은 간단하다. Generate 버튼을 입력하였을 때, 단어 유사도 비교가 끝난 세 개의 단어를 기반으로 비밀구절을 생성하여 보여준다. 보여주는 단어는 원본, 발음과 유사한 문자로의 치환 과정이 끝난 단어와 최종 결과물이다. 루프 발생 시의 자원 낭비 및 전체 코드 실행 시간 단축을 위해, 어휘를 DB에 탑재한다. 이후, 탑재된 단어를 발음에 직결되는 단어로 변환하여 추가적으로 저장한다. 변환 예시는 [표 1]과 같다. 알파벳과 숫자의 발음이 같을 경우, 숫자의 발음을 더 우선시하여 치환한다. 치환 완료된 단어 역시 DB에 저장하며, 암호 복잡도 계산을 편리하게 하기 위해 별도의 구별용 필드를 생성하여 저장한다. 숫자가 포함된 경우 1, 대문자가 포함된 경우 2의 값을 저장하며 숫자와 대문자 모두 포함된 경우 3을 저장한다. 이후, 추출된 단어의 구별 필드 값의 총합이 4 이상, OR 비트 연산 결과가 3일 경우에만 다음으로 진행한다.

어휘	발음 기반	Qwerty화	구별값
콜라	콜라	zhffk	0
누렁이	누렁2	snfjd2	1
로가티오	로가t5	fhrkt5	1
싸리개비	싸리개b	Tkflrob	2
재떨이	재떨2	woEjf2	3

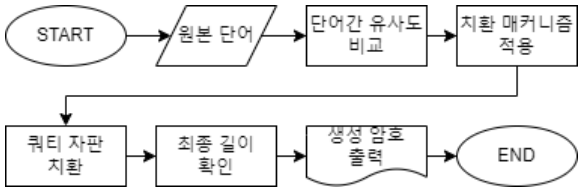
[표 1]

최종적으로 변환된 암호를 하나의 string으로 합한 뒤, 암호 정책에 부합되는지 확인한다. 암호 정책은 Windows의 기본 암호 정책을 기반으로 진행할 예정이며, 해당 정책의 캡쳐본을 [그림 1]로 첨부하였다. 해당 프로세스에 관한 순서도는 마지막의 [그림 2]와 같이 표현하였다.

현재까지의 예상되는 문제점으로는, 특수기호 사용이 필수인 환경에서 본 시스템의 적용이 어려울 수 있다는 점이다. 그 경우, 추가적으로 추출된 단어의 숫자를 쿼티 자판에 맞추어 특수 기호로 변환하는 방안이나, 단어와 단어 사이에 임의의 특수기호 삽입을 고려한다.



[그림 1]



[그림 2]

3.2. 요구사항

3.2.1 NLP 모델에 대한 요구사항

코사인 유사도를 기반으로 단어 간의 유사도를 비교하는 딥러닝 모델인 Word2Vec, 혹은 사전 학습이 완료되어, 추가 학습만을 진행해도 무방한 BERT를 이용하여 학습을 진행할 계획이다. 기

존에 제기된 문제점으로, Word2Vec은 동음이의어나 다의어를 별개의 단어가 아닌 하나의 단어로 취급하기 때문에 유사도 비교의 의미가 다소 희석된다.[4] 따라서, 프로젝트 진행 시간의 여유가 있다면, 동일한 자료로 모델을 학습시켜 프로젝트에 더 적합한 모델을 기반으로 단어 유사도 비교를 진행한다. 과적합 방지를 위해 수집 데이터를 학습, 검증, 시험 데이터로 나누기 때문에, 학습 데이터의 과소적합을 방지하기 위해, 가능한 많은 학습 데이터를 수집하여야 한다. 수집된 데이터의 양이 충분한지를 확인하기 위해, 동일 자료를 기반으로 최소한 2개 이상의 모델을 학습 후 확인하여야 한다.

3.2.2. 암호 생성기에 대한 요구사항

가능한 한 암호 복잡도 미달로 인한 재귀를 방지할 수 있도록 설계되어야 한다. 앞서 언급한 단어 후보군 사전 변환 등의 전처리를 고려하여 일정 수준의 암호 복잡성을 보장받을 수 있도록 해야 한다. UI는 최대한 간단하게 표현되어야 하며, 비밀구절 출력 창은 클립보드 복사 기능을 지원하거나, 최소한 드래그가 가능하도록 구현해야 한다.

4. 구현 관련 정보 및 결과

본 연구를 위해 설정한 시스템 환경 및 언어, 클라우드 등의 요소는 [표 2]와 같이 정리하였다.

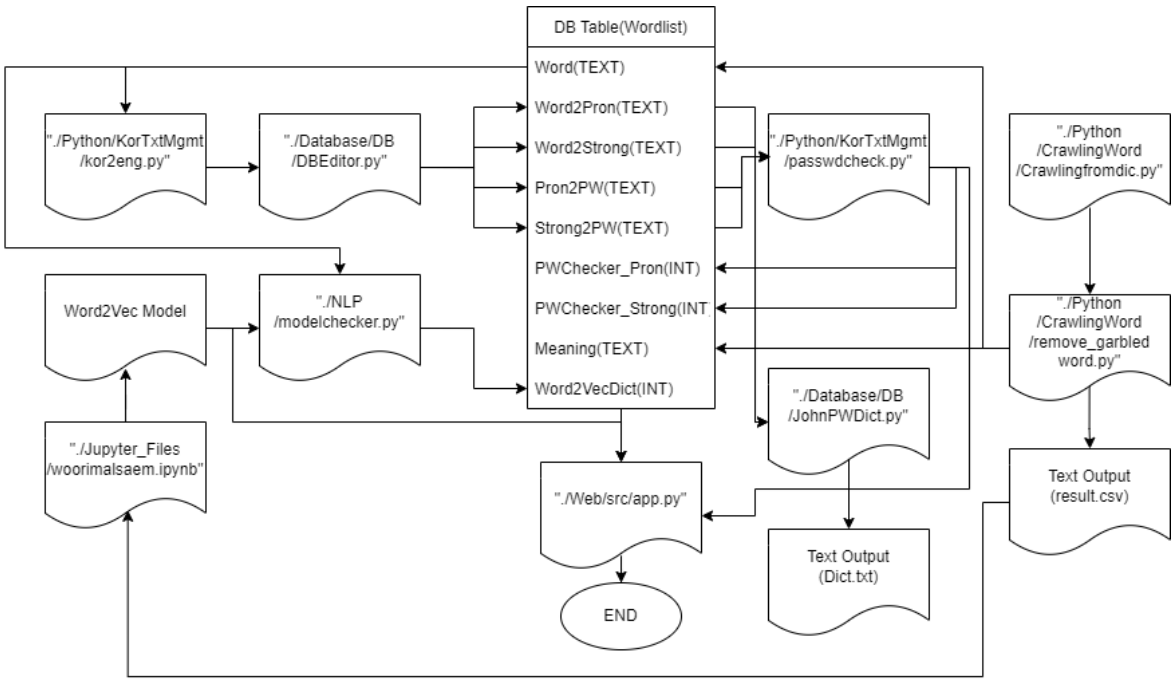
CPU	AMD Ryzen 3400G 3.7GHz
RAM	DDR4-25600 16GB x 2
OS	Windows 10 Education
Database	SQLite3
Language & Framework	Python 3.9 (Word2Vec comp.) / Python 3.10 (Latest) HTML & CSS (Powered by Flask)
추가기능	Compatibility Layer (WSL2 Kali Linux) Cloud (Google Colaboratory)

[표 2]

Word2Vec 모델 생성에는 Gensim 라이브러리를 활용하였으며, 데이터로는 2022년 3월 기준의 우리말샘 사전 데이터 및 한글 위키피디아 문서를 사용하였다. 해당 데이터를 자연어 처리하기 위한 전처리기로서 형태소 분류기는 Mecab에 한국어 형태소 데이터를 탑재하여 활용하였다. 생성 암호 강도를 측정하기 위해 Windows 10부터 제공되는 하위 시스템 기능인 WSL2를 활용하여 Windows 상에 Kali Linux를 설치하여 활용하였다. 추가로, 단시간에 최대한 많은 모델 학습 결과를 확보하기 위해 우리말샘 사전 데이터는 Google의 Colaboratory 상에 탑재하여 학습을 진행하

였다.

본 프로젝트의 DB는 단순히 단어 정보를 불러오기만 하면 되기 때문에 CSV 등의 스프레드시트를 활용해도 무방했으나, 데이터 접근 속도 및 관리의 용이성, 프로젝트 규모 측면에서 적합한 SQLite3를 채택하였다.



[그림 3]

Github에 게시한 코드의 구성을 [그림 3]과 같이 개괄적으로 표현하였다. 이 중, 사용자가 암호 생성기 서비스를 이용하는 데에 직접적으로 사용되는 코드 및 구성요소는 Word2Vec 모델, 데이터베이스, passwdcheck.py 및 app.py이다. 이외의 코드는 데모 서비스 단계에서 사용한 코드 혹은 최종 단계에서 생략되거나 용도폐기된 코드다. [그림 3]에 표기한 코드 파일의 최소 1/3은 DB 파일과의 상호작용이 필요하며, 최소한의 기능 및 코드 탑재만을 필요로 하는 로컬 배포판을 제작할 경우, Word2Vec 생성 모델, app.py 및 웹 구성요소, passwdcheck.py만을 필요로 할 것으로 전망한다. 물론 네트워크 환경에 해당 서비스를 호스팅할 수 있다면, 서비스 주소만을 배포하는 것으로 구성 파일 접근 보안만을 신경쓰면 될 것으로 전망한다.

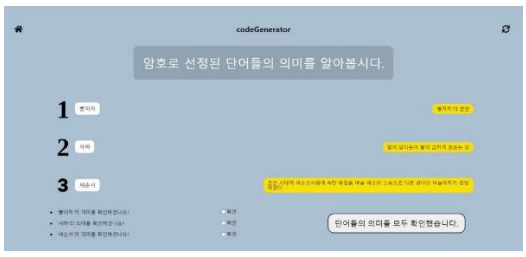
app.py 코드의 개괄적 구성은 다음과 같다.

app.py
calculateWordSimilarity(word_01, word_02)
generateRandomPW(num)
hasNumber(stringVal)
index() - home.html 호출
result() - check() 결과 따라 final()로 Redirect
check() - special()로부터 GET한 암호 점검 기능
mean() - 암호로 사용할 단어 추출, 유사도 체크
choice() - mean() 추출 단어 된소리화 여부 선택
special() - choice() 결과 단어 특수기호 치환 선택
final() - result()를 통해 최종 암호 출력

[그림 4]

calculateWordSimilarity(word_01, word_02) 함수는 generateRandomPW(num) 함수 및 Word2Vec 모델의 영향을 받는다. Word2Vec 모델 학습 결과 전반적인 유사도가 80% 이상일 때 실생활에서도 충분히 유사함을 감지할 수 있는 단어가 추출되며, DB에 수록한 단어가 20만여 개임을 감안하여, 평균 유사도가 83.3%p를 초과할 경우에만 재추출을 수행하도록 설정하였다. 50%p로 설정하였을 때의 평균 재추출 횟수가 10회였던 점을 감안했을 때 유의미한 결과다.

[그림 5] – home.html (http://localhost:5000)



[그림 6] – mean.html (http://localhost:5000/mean/)

[그림 7] – choice.html (http://localhost:5000/choice?check_info=firstPw&check_info=secondPw&check_info=thirdPw)

[그림 8] special_symbol.html (http://localhost:5000/special/)

[그림 9] check.html
(http://localhost:5000/check/)

[그림 10] final.html
(http://localhost:5000/final/)

서비스의 구성 및 동작 자체는 매우 간단하다. 암호를 2회 입력하며 숙지하는 단계를 제외하고는 대부분의 작업은 마우스 조작만으로 가능하다. 암호 길이 제한이 없는 경우 혹은 숙지 난이도가 높아지더라도 강력한 암호를 생성해야만 하는 경우라면 [그림 5] 단계에서 "필터 없음" 혹은 "5글자 이하"를 선택하여 긴 글자가 필터링되지 않도록 설정한다. "필터 없음"을 설정할 경우 길이 20 이하의 단어를 추출하도록 설정하였다.

그렇게 추출된 단어는 국어사전을 기반으로 추출되었기 때문에 난수화 암호에 비해서는 친숙한 형태이지만, 일상생활에서 자주 사용되지 않을 가능성이 높다. 따라서, 해당 단어를 보다 용이하게 숙지하기 위해 추출된 단어들의 의미를 [그림 6] 기준으로 우측에 표기하였다.

추출된 단어는 된소리화(ㄱ, ㄷ, ㅂ, ㅅ, ㅈ를 쌍자음화하는 것)가 가능할 경우, 사용자 기호에 따라 치환할 수 있도록 [그림 7]과 같이 선택지를 제공한다. 다만, 규칙의 통일성 및 암호 강도 향상을 위해 발음 기반 치환은 기본값으로 적용한다.

[그림 8]은 치환 규칙을 적용한 단어에 추가적으로 특수기호 치환을 적용할 부분을 선택한다. 현재는 숫자만을 특수기호로 전환하는 기능만을 제공한다. 그렇게 완성된 암호는 [그림 9]에 표시된 2칸의 입력 창에 각각 한 번씩 입력하며 암호의 숙지 용이성을 한 번 더 끌어올린다. 최종적으로 [그림 10]과 같은 형태로 암호 구성 단어 간의 의미적 유사도의 평균을 수치화하여 표기하고, 하단의 그래프를 통해 개략적인 암호 강도를 확인할 수 있도록 한다.

본 서비스를 구축하는 데에 사용된 Word2Vec 모델에 관한 구축 방향 및 생성 암호 검증에 관한 내용은 별도로 첨부한 논문("CD1_DLN(출판논문).pdf")에 상세히 기록하였다. 요약한다면, Word2Vec을 통해 학습시킨 모델의 결과는 단어사전 탑재율에 한정해서는 단어사전만을 이용하여

학습시킨 결과가 프로젝트에 응용하기에 가장 적합하였다. 전체적인 탑재율이 낮아지는 결과를 낳았는데, 이는 한국어 형태소 분류기가 신조어 및 옛우리말 등의 자주 쓰이지 않는 단어에 대해 대응하지 못하는 형태소 분류기 DB 기반 분류 방식을 채택하고 있기 때문이다. 또한, 암호 강도 검증의 경우, John the Ripper는 암호사전 내의 단어를 변형하는 기능은 제공하더라도 단어를 조합하는 기능은 제공하고 있지 않다. 따라서, 수록한 460,100개의 단어로 조합할 수 있는 모든 경우의 수를 사용하여 텍스트 파일을 생성한다고 한다면 그 용량은 압축 없이 200PB 정도로 추산되며, 압축을 거친다고 하더라도 일반적인 환경에서 크래킹 환경을 구축하는 데에는 다소 무리가 있을 것으로 추정된다.

5. 결론 및 기대효과

암호 복잡도 정책 및 높은 수준의 암호 강도를 만족하면서, 동시에 숙지 및 배포가 용이한 암호를 손쉽게 생성하여 단체 계정 생성 및 배포 시의 초기 암호 설정 및 임시비밀번호 지급 혹은 임시 관리자 권한 부여 시 최소한의 안전성 확보할 수 있을 것으로 기대한다.

장기적으로는, 생성되는 비밀구절이 일정한 패턴을 따르기 때문에, 비밀구절을 생성할 때 해당 프로세스를 이용한다는 사실이 노출된다면, 암호의 파훼가 무작위 대입 공격보다는 더 용이하다는 단점이 있다. 하지만 해당 수준의 비밀구절을 적용해도 무방한 시스템(암호 만료 주기가 짧거나, 부여된 권한이 적은 일반 계정 등)에 해당 비밀구절 생성 시스템을 이용한다면, 해당 암호 생성기의 단점을 제도적으로 보완할 수 있다고 전망한다.

향후 연구는 해당 연구의 역으로서 암호를 입력하였을 때 쿼티 기반 암호의 의미적 패턴을 분석하여 암호 강도 측정에 활용하는 방향으로의 진행을 고려할 수 있다. 추가로, 한국어 자연어처리 모델이 신조어나 전문용어에 더욱 잘 대응할 수 있도록 딥러닝 등의 기술을 도입한 유연성 높은 전처리기의 연구가 필요할 것으로 전망한다.

6. 참고문헌

1) 「Oregon FBI Tech Tuesday: Building a Digital Defense with passwords」, 『FBI』, 2020년 2월 18일.

2) 「온라인 비밀번호, '이렇게 지어야' 더 효과적」, 『IT조선』, 2020년 2월 26일.

3) 김호용·이민호·서동민, 「우리말샘 사전을 이용한 단어 의미 유사도 측정 모델 개발」, 『종합학술대회 논문집』 2018년도 춘계, 한국콘텐츠학회, 2018.

4) 이치훈·이연지·이동희,「사전 학습된 한국어 BERT의 전이학습을 통한 한국어 기계독해 성능개선에 관한 연구」,『한국IT서비스학회지』제19권 제5호, 한국IT서비스학회, 2020.

5) 박상길·신명철,「Sent2Vec 문장 임베딩을 통한 한국어 유사 문장 판별 구현」,『한글 및 한국어 정보처리 학술대회』2018년도 제30회, 한국정보과학회언어공학연구회, 2018.

6)「미 국립표준기술연구소, 패스워드 관리에 대한 권고사항」,『금융보안원』, 2017.

7) 임근영·조영복,「딥러닝과 Char2Vec을 이용한 문장 유사도 판별」,『한국정보통신학회논문지』제22권 제10호, 한국정보통신학회, 2018.