컴퓨터 비전을 활용한 문자 입력패턴 시각화/분석기술 연구

한국정보보호학회
2022년 2월 24일 ~ 2022년 11월 23일

연구책임자
소속: 중앙대학교
직위: 조교수
성명: 최종원

컴퓨터 비전을 활용한 문자 입력패턴 시각화/분석기술 연구

본 보고서를 과제의 최종 결과물로 제출함

2023년 11월

중앙대학교 첨단영상대학원
영상학과 조교수
최 종 원

목 차

1. 연구개요

가. 연구의 중요성 및 목표

(1) 연구 중요성

- 사용자에 따른 자소별 활용 빈도가 차이나며, 이는 사용자를 특정지을 수 있는 지표가 될 수 있음
- 사용자 특성을 기반으로 새로운 내용 입력 시 타이핑 스타일 예측 가능 여부 확인 필요
- 자소별 활용 빈도에 따른 개인 특정화로 인한 개인 정보 누출 문제를 막기 위한 자동 보안 시스템 구축

(2) 연구 목표

- 자소 타이핑 패턴에 대한 인공지능 인식 기술 활용 및 분석
- 사용자 특성을 이용하여 맞춤형 개인 정보 보안을 위한 타이핑 추천 시스템 구축
- 사용자 특성 (게시글 등)에 기반한 개인화 및 분류 알고리즘 적용 방법론 연구

(3) 연구 주요내용

- 사용자별 특성화된 타이핑 데이터 분류 학습을 위한 데이터 수집
- 사용자별 타이핑 패턴을 시각 데이터로 표현하는 방법론 개발
- 시각화된 타이핑 패턴을 기반 유사도 및 거리를 계산할 수 있는 딥러닝 방법론 개발
- 시각화 데이터 및 사용자 특성을 활용한 개인화 및 분류 알고리즘 테스트 수행
- 시각화 타이핑 패턴 기반 타이핑 추천 시스템 개발 및 개인 정보 보호 시스템 연구

나. 관련 연구 동향

(1) 국내외 연구기술 개발 현황

(가) 키보드 설계를 위한 인접-오타 타자 분석 - 한국외국어대학교[1)]

- 소프트 키보드 (터치스크린 키보드)의 물리적 한계로 인한 문자 입력 속도와 오타 발생 확률 측면의 사용성을 높이기 위해 동적 적응 방법론 기반의 알고리즘 제안

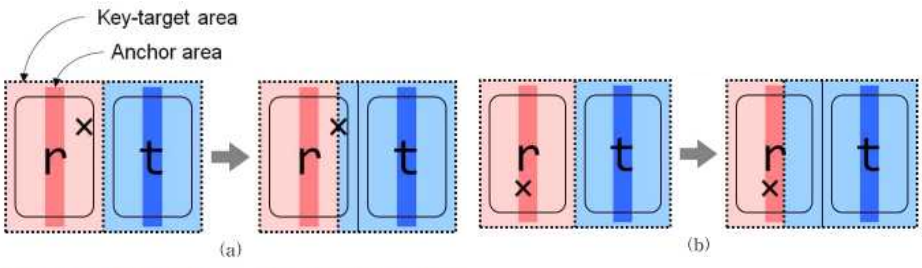


그림 1. 키보드 오타 분석을 위한 동적 적응 알고리즘 설명 [1]

- 키 입력 시계열 데이터를 분석하여 BS (backspace)키가 눌린 위치를 중심으로 오타와 원래 입력하려 했던 문자 추출 후 오타와 인접 오타 구분 수행

1) 고석훈, “인접-오타를 이용한 소프트 키보드의 동적 적응 연구”, Journal of Korea Multimedia Society, Vol. 21. No. 11., Nov. 2018 (pp. 1263-1270)

- 정타에 대해서만 적응을 수행하는 기존의 동적 적응 방법과 인접 오타를 이용한 동적 적응 방법을 사용하여 실험적 분석 수행
- 사용자의 입력으로부터 추출한 오타 간의 관계 정보를 이용하여 키보드 인식 범위 수정을 통해 기존 연구에 비해 25%의 사용성 증가를 보여줌

(나) TypeVis: Visualization of Keystrokes and Typing Patterns - University of Debrecen[2)]

- 사용자의 타이핑 스타일을 시각화하는 방법을 제안함
- 시계열 분석 기법을 바탕으로 사용자가 키보드를 누르고 떼는 시간 차이를 계산하여 시각화 작업 수행

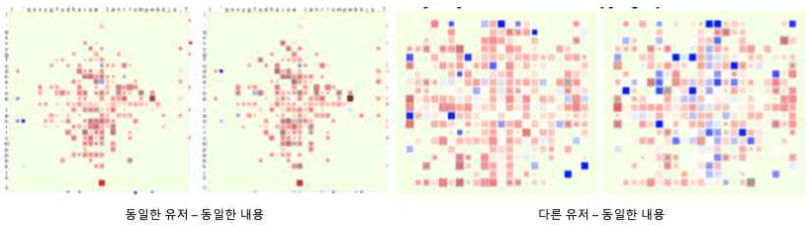


그림 2. 사용자의 키 입력 시간 측정을 통해 시각화한 이미지 [2]

- 시각화된 자료를 바탕으로 사용자의 습관과 같은 특성을 시각화함

(다) 키보드 자판 설계를 위한 키보드 레이아웃 설계 - 독일 Neo-Layout[3)]

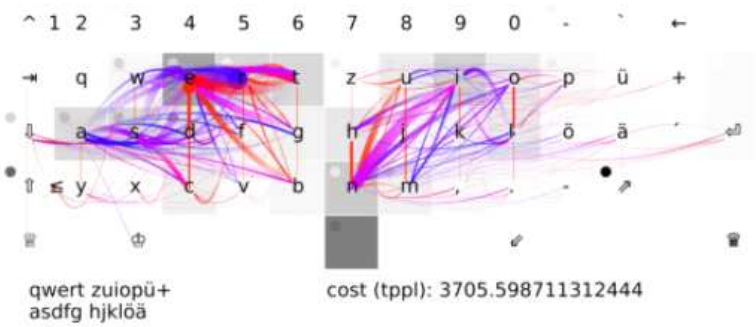


그림 3. Neo-Layout을 활용한 타이핑 분석 결과 [2]

- 키보드 타이핑 비용과 오타 입력 개선을 위해 오타율을 측정하여 기존 키보드 레이아웃 간의 비교 진행
- 키보드 타이핑 비용은 입력하는 문자 또는 키보드 배열에 따른 같은 손가락 사용 빈도, 연속되는 입력 사용 빈도 등을 기준으로 측정
- 비용을 최소화하는 학습이 가능하도록 학습 가능한 파라미터를 설정하여 레이아웃을 개선하였고 블라인드 타이핑 테스트를 통해 성능 실험 진행
- 개발한 키보드 레이아웃의 타이핑 비용 기준으로 성능 측정 결과 평균 결과값 11로 최적화된 레이아웃 달성

2) Enyedi, Kinga, and Roland Kunkli. "TypeVis: Visualization of Keystrokes and Typing Patterns based on Time Series Analysis." *VISIGRAPP (3: IVAPP)*. 2019.
3) https://www.neo-layout.org/

2. 연구수행내용 및 결과

가. 자소 타이핑 데이터의 시각화 및 분류 알고리즘

(1) COMB dataset[4] - 시각화 활용 데이터

- 기존 대다수의 연구는 keystroke dynamics를 기반으로 진행되어 본 과제의 방향성과 다소 차이가 있음
- keystroke dynamics는 사용자가 키보드를 입력할 때 자판을 누르고 떼고, 다음 자판을 누르는 시간 등을 수집하여 어떤 습관이나 특성으로 자판을 입력하는지에 대한 연구를 의미함
- 일반적으로 사용자 별로 키보드 자판을 누르고 떼는 시간이 다른 것을 기반으로 한 이상치 탐지 작업이 수행되어옴
- 본 과제에서는 사용자가 입력한 자소 자체를 기반으로 시각화를 수행하기 때문에 keystroke dynamics 데이터셋을 사용하지 않음

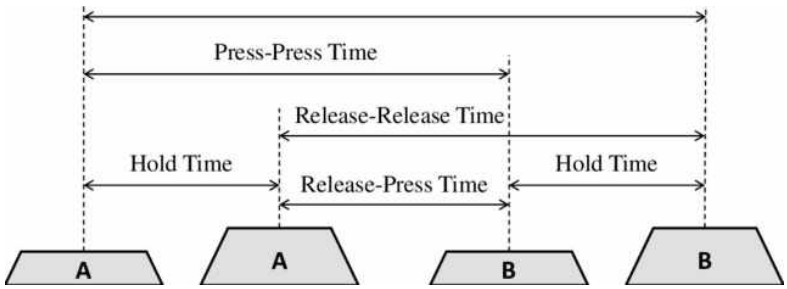


그림 4. Keystorke dynamics 수집 방법

- 앞서 조사한 선행 연구들을 바탕으로 사용자를 특정할 수 있는 데이터를 사용해 시각화하고 입력 문자만으로 사용자 구별이 가능한지 시각화 및 실험을 수행함
- 사용자가 과거에 사용하거나 작성한 문자열을 구축한 데이터셋이 현재까지 존재하지 않기 때문에 웹사이트 상에서의 댓글 이력을 수집하여 시각화를 시도해보았으나 특정 분야에서 매우 유사한 형태의 문자열을 사용하기 때문에 사용이 어려운 것을 발견함
- 동일한 사용자가 과거에 사용한 자소 데이터가 시각화 및 인공지능 모델을 학습시킬 수 있는 충분한 양이 있는 데이터 수집 수행

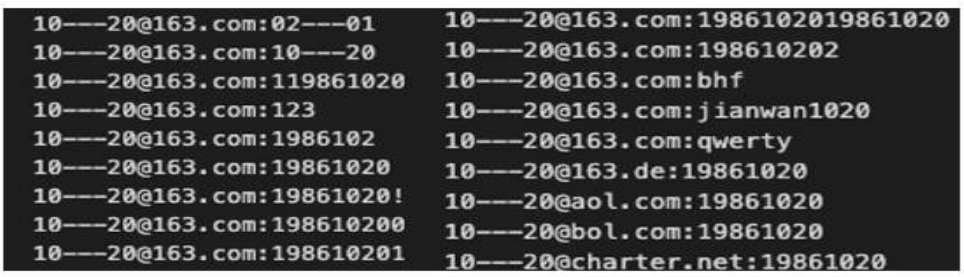


그림 5. COMB dataset의 데이터 예시 [3]

4) https://cybernews.com/news/largest-compilation-of-emails-and-passwords-leaked-free/

- COMB는 페이스북, 인스타 등에서 동일한 계정을 가진 사용자의 유출된 비밀번호의 모음으로 그림 5에서 볼 수 있듯이 한 사용자가 여러 웹사이트에서 사용한 비밀번호가 수집되어 있음
- 전체 데이터는 약 3백만개로 사용자 아이디의 알파벳 기준 및 트리 형태로 데이터가 구성되어 있음
- 다크웹에서 공개된 데이터로 파일 전송 P2P 기술인 토렌트에서 데이터 수집 진행
- 아이디 별로 수집된 문자열의 개수가 상이하며 100개 이상 동일한 아이디에서 수집된 문자열의 개수는 467,148개이고 100개 이상 수집된 아이디의 개수는 7,580개이므로 많은 사용자들이 사용할 것으로 예상되는 간단한 아이디의 경우 1,000개 이상 수집된 경우도 있음
- 앞서 언급한 바와 같이 비정상적으로 동일 아이디에 대해 비밀번호가 수집된 경우 인공지능 모델을 학습시키는데 과도한 편향을 줄 수 있어서 이를 제외함

(2) 시각화 방법론

- 계정 별 비밀번호 입력 패턴을 사용해 시각화 수행 했으며 동일한 계정에서 비밀번호가 100개 이상 수집된 계정을 대상으로 수행함
- 영문, 숫자 및 보편적으로 사용되는 특수기회 외의 다른 문자를 포함한 경우의 비밀번호는 제외함
- 다양한 선행 연구의 시각화 기법들을 참조하였고 비밀번호 데이터를 기반으로 시각화를 했을 때 해당 사용자의 특성이 실제 시각적으로 두드러지는 차이가 있는 방법론을 연구함

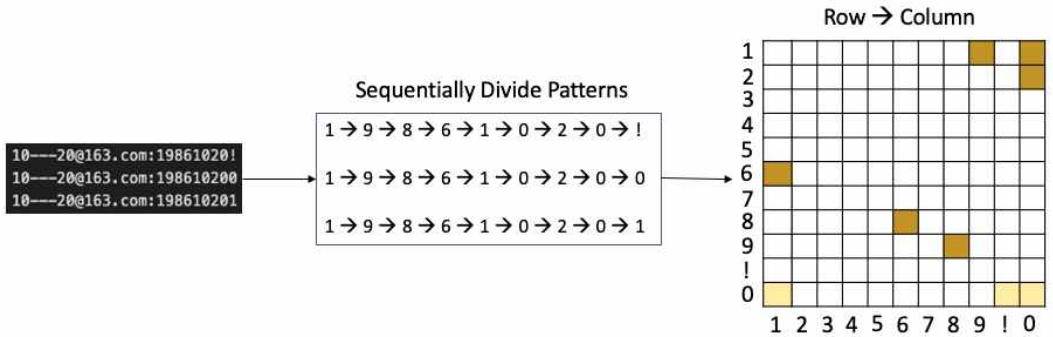


그림 6. 비밀번호 시각화 방법론

- 그림 6에서 볼 수 있듯이 비밀번호 입력 데이터는 시계열 자료의 형태를 띄고 있기 때문에 이러한 특징을 시퀀스 형태로 해당 문자열에서 다음 문자열로 이동하는 것을 시각화 작업을 수행함
- 알파벳, 숫자, 특수문자 100개를 기준으로 패턴 입력 순서에 따라 100x100행렬로 표현함
- 그림 6에서 진한 노란색으로 표현이 되는 것처럼 사용자가 동일한 문자열을 작성한 경우 해당 부분에서 시각적으로 더 강조될 수 있도록 표현함
- 개인 특정 분류 모델 및 생성 모델 학습을 위해 사용자가 사용한 문자열이 겹치지 않도록 생성함

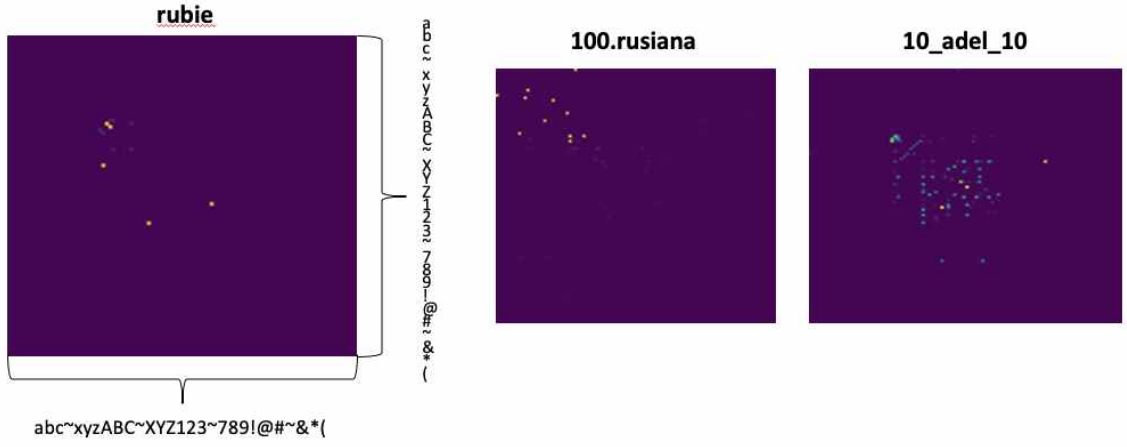


그림 7. 시각화 결과 예시

- 그림 7은 그림 6에서 설명한 방식으로 세 명의 사용자를 시각화한 이미지임
- 그림 7에서 볼 수 있듯이 각 사용자들이 자주 사용하던 문자열 패턴에 따라 해당 구역에서 색이 달라진 것을 확인할 수 있으며 다양한 문자열 패턴을 사용한 10_adel_10 아이디의 사용자는 옅은 초록색 부분을 많이 띔

(3) 시각화 데이터 기반 개인 특정 학습 방법론

(가) 분류 모델 접근 방법

- 앞서 시각화한 데이터를 기반으로 사용자 인공지능 분류 모델을 구축하는 것을 목표로함
- 시각화 데이터의 해당 사용자가 누구인지 분류하는 작업을 통해 시각화 방법론의 정합성을 확인할 수 있음
- 대표적으로 알려진 이미지 분류 데이터셋인 ImageNet, MNIST, CIFAR 등이 본 과제에서 수행한 시각화 이미지의 사용자 분류 작업과 연관이 있어서 해당 데이터셋에서 사용된 인공지능 모델을 적용함

(나) 합성곱 신경망 (Convolutional Neural network, CNN) 접근

- 합성곱 신경망은 인간의 시신경 구조를 모방한 기술로 특징맵을 생성하는 필터까지도 학습이 가능해 컴퓨터 비전 분야에서 우수한 성능을 보여줌
- 데이터를 직접 학습하고 패턴을 사용해 이미지를 분류함
- 이미지의 공간 정보를 유지한 채 학습을 하게 하는 모델로 이미지를 1차원이 아닌 2차원 형태 그대로 연산을 수행함
- 이미지를 인식하기 위해 패턴을 찾는데 특히 유용하여 본 과제에서 개인 특정 학습 방법론에 적용하여 사용자를 분류할 수 있는 모델을 구축함
- 합성곱 신경망은 이미지의 특징을 추출하는 부분과 클래스를 분류하는 부분으로 나눌 수 있음
- 특징 추출 영역은 convolution layer와 pooling layer를 여러 겹 쌓는 형태로 전자는 입력 데이터에 필터를 적용 한 후 활성화 함수를 반영하는 필수 요소이고 pooling layer는 추출한 특징의 크기도 줄이고 특정 부분을 강조할 수 있는 역할을 수행함.

- 클래스를 분류하는 부분은 완전 연결 계층 (fully-connected layer)는 추출된 특징으로 이미지를 분류하는 작업을 수행함.

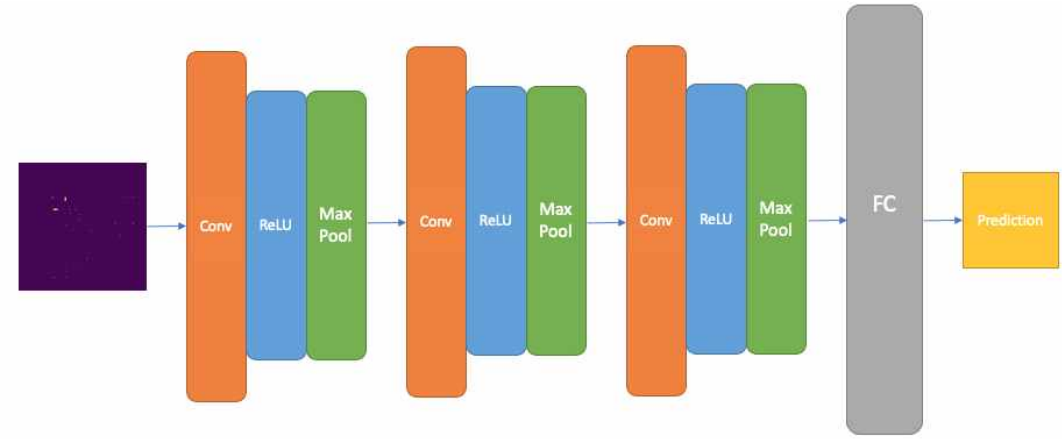


그림 8. 합성곱 신경망 분류 모델 구조

- 그림 8에서 보이는 바와 같이 깊지 않은 합성곱 신경망을 구성함
- 많이 알려진 ResNet, VGG, Inception 모델을 사용하지 않은 것으로 해당 모델들은 ImageNet에서 볼 수 있는 강아지, 고양이 등 다양한 이미지를 학습시키기 위한 깊은 모델임. 따라서 본 과제에서 시각화한 데이터는 픽셀 간의 값 차이가 두드러지게 나지 않기 때문에 그림 8과 같이 얕은 신경망을 구축함
- 합성곱 신경망을 구축하여 간단히 실험을 수행한 결과 테스트셋에서의 정확도는 14.5%에 불과했음
- 이는 시각화한 데이터의 경우 충분한 강도와 양의 픽셀 값을 지니고 있지 않아서 합성곱 신경망 모델의 성능이 낮은 것으로 확인됨
- 주변 픽셀 값을 함께 aggregate하는 합성곱 연산의 특성에 적합하지 않음
- 각 픽셀의 위치 정보가 중요하나 합성곱 연산은 상대적 좌표 정보만을 고려한다는 단점이 있음

(다) 다층 퍼셉트론 (multi-layer perceptron, MLP)

- (나) 절에서 합성곱 신경망을 구축하여 실험을 수행했으나 원하는 수준의 정확도를 달성할 수 없었음
- 주변 값들의 영향을 적게 받는 모델인 다층 퍼셉트론 (multi-layer perceptron, MLP)를 적용해 실험을 수행함
- Fashion minst 데이터셋에서 이미지를 분류하는 인공지능 모델로 널리 알려져 있음
- 다층 퍼셉트론을 적용하기 위해 2차원 벡터의 행렬을 1차원 배열로 평탄화하여 다층 퍼셉트론의 입력으로 사용하게 되는데 좋은 성능을 보여주고 있으며 본 과제에서 시각화한 데이터 또한 2차원 벡터의 행렬이기 때문에 적용함
- 다층 퍼셉트론은 여러 개의 퍼셉트론 뉴런을 여러 층으로 쌓은 다층신경망 구조로 입력층과 출력층 사이에 하나 이상의 은닉층을 가지고 있는 신경망임
- 인접한 두 층의 뉴런 간에는 완전 연결이 되며 XOR과 같은 비선형 분리가 가능한 결

정선 생성 가능함

- 다만 인접한 층의 퍼셉트론 간의 연결은 있어도 같은 층의 퍼셉트론끼리의 연결은 없으며 지나간 층으로 다시 연결되는 피드백도 없음
- 다층 퍼셉트론은 층의 개수 (depth)와 각 층의 크기 (width)로 결정됨

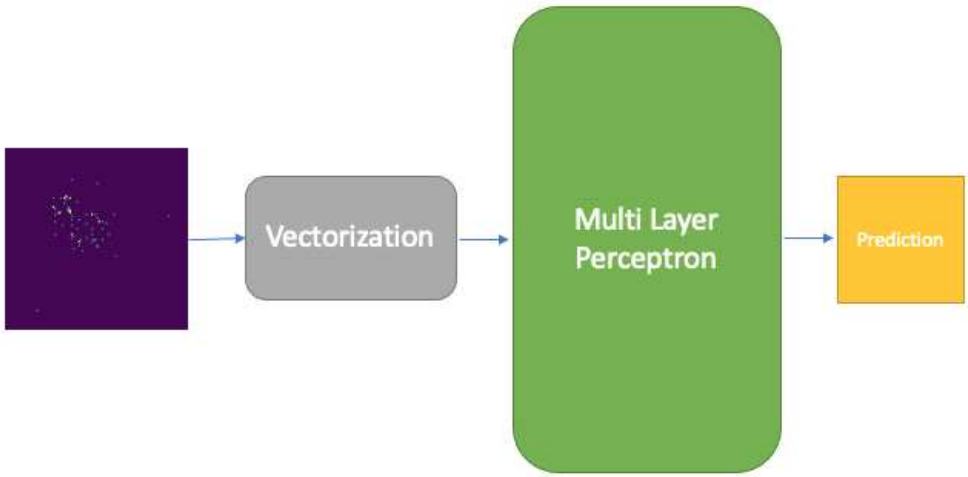


그림 9. 다층 퍼셉트론 분류 모델 구조

- 그림 9에서 보는 바와 같이 fashion mnist에서 적용했던 방식처럼 평탄화 작업 즉 벡터화를 수행하게 되며 1차원 배열로 변환된 이미지를 다층 퍼셉트론의 레이어의 입력으로 사용함
- 추가적으로 시각 이미지 간의 평균과 분산이 모두 다르기 때문에 표준화 작업을 벡터화 이후 추가적으로 수행함
- 앞서 언급한 다층 퍼셉트론의 층의 개수와 각 층의 크기는 정확도 평가에서 비교 실험을 통해 최적의 값을 구함

(4) 분류 모델 성능 및 결과

- 100명의 사용자를 예측하는 모델을 한 층의 fully-connected layer로 구성된 MLP로 학습 수행
- 시각화 데이터는 사용자별 30개의 이미지를 학습 및 테스트셋으로 8:2 비율로 나누며 학습과 테스트셋에 시각화에 사용된 패스워드는 각기 다른 패스워드로 구성됨
- 비용 함수로는 다중분류 모델에 대표적으로 사용되는 cross-entropy를 적용함
- 총 100 epochs를 수행하였으며 테스트셋의 정확도가 5번 이상 향상되지 않으면 학습을 멈추는 형태를 사용했음
- Accuracy 평가지표를 활용해 모델의 성능 측정하였으며 올바르게 예측된 데이터의 수를 전체 데이터의 수로 나눈 값을 의미함
- 모든 모델은 optimizer로 Adam, 활성화 함수로는 ReLU를 동일하게 적용하였음
- 표 1에서 보는 바와 같이 합성곱 신경망과 다층 퍼셉트론 간의 정확도 성능 차이가 확연하게 나는 것을 확인할 수 있음

Method	Accuracy	Hidden_size	학습률	L2규제
CNN	14.5%	100	0.01	0.001
MLP	78.27%	100	0.1	0.01
MLP	88.67%	100	0.01	0.01
MLP	89.07%	100	0.01	0.001
MLP	88.67%	100	0.01	0.0001
MLP	89.20%	100	0.001	0.001
MLP	67.20%	50	0.001	0.001
MLP	79.37%	120	0.001	0.001

표 1. 분류 네트워크의 예측 성능 평가표

- 이는 학습에 사용하는 시각화 이미지가 3차원의 컬러 이미지가 아니며 이미지 내의 픽셀 간의 간격이나 각 픽셀 값의 크기가 작기 때문으로 해석됨
- 다층 퍼셉트론 모델의 은닉층의 깊이를 1개로 학습을 했음에도 불구하고 표 1과 같이 정확도 89.20%를 획득할 수 있었음
- 또한, 은닉층의 깊이를 1개로 했음에도 불구하고 유의미한 성능을 보여주었는데 이는 시각화 이미지가 사용자의 특성을 잘 표현하는 것으로 볼 수 있음
- 은닉층의 넓이는 100을 기준으로 크게 혹은 작게 했음에도 불구하고 100이 가장 좋은 정확도를 보여주었음
- 본 절에서 시각화한 이미지를 인공지능 분류 모델을 통해 사용자 분류를 수행하고 해당 결과를 바탕으로 시각화 이미지가 사용자의 자소 타이핑 패턴을 잘 보여주는 것으로 해석할 수 있음

(5) 알고리즘 코드 실행환경 및 사용방법

(가) 구동 하드웨어 및 소프트웨어 환경

- OS: Ubuntu 16.04
- CPU: Intel Xeon Gold 5220 (8 cores, 2.2 Ghz)
- GPU: Tesla V100-SXM2-32GB
- Python 3.6, Pytorch 1.7.1

(나) 코드 실행 방법

- 해당 폴더로 이동 후 파이썬 패키지 설치

```
$ cd password
$ pip install requirements.txt
```

- 사용자 비밀번호 데이터를 시각화하는 코드 실행
- 해당 코드를 실행할 경우 password/data/train, password/data/test 폴더 안에 이미지가 생성됨
- 미리 시각화된 데이터를 활용할 경우 코드 실행하지 않아도 됨

```
$ cd password/visualization
$ python vis.py
```

- 분류 작업 수행 코드 실행
- argument로 —train을 True 또는 False를 사용하여 미리 학습된 모델을 이용해 테스트를 진행할지 새로 학습하여 테스트를 진행할지 선택할 수 있음
- 학습된 모델은 password/classfication/ckpt 안에 저장되어 있음

```
$ cd password/classification
$ python class.py --train True or False
```

나. 자소 시각화 데이터 기반 응용 알고리즘

(1) 비밀번호 생성모델 학습 방법론

- 사용자의 비밀번호 사용 패턴을 기반으로 비밀번호 생성모델을 구축하여 사용자의 과거 사용 비밀번호를 예측하는 방법론 개발을 수행함
- 현재 적대적 신경망 (generative adversarial networks, GAN)은 이미지 생성 분야에서 압도적인 성능을 보여주고 있으며 입력 데이터를 기반으로 새로운 이미지를 만들어 내는데 탁월한 능력을 지니고 있음
- 비밀번호 생성모델이 사용자의 과거 비밀번호를 맞출 경우, 해당 사용자는 일정한 패턴의 비밀번호를 사용하는 것이며 새로운 비밀번호를 사용하는 것을 추천할 수 있음

(가) 적대적 신경망 (generative adversarial networks, GAN)

- 적대적 신경망과 같은 생성 모델은 특정한 확률분포를 갖는 학습 데이터를 이용해 학습을 시키면 이것과 유사한 분포를 갖는 데이터 혹은 샘플을 생성할 수 있는 모델을 말함
- 학습 데이터를 이용해 학습만 시키며 정답을 붙여주지 않더라도 스스로 데이터 속에 숨은 중요한 정보를 찾아내고 새로운 데이터를 생성하는 것임
- 생성망 (generator)와 판별망 (discriminator)를 경쟁적으로 학습을 시켜 진짜인지 혹은 가찌인지 구별하기 힘든 수준까지 발전시키는 모델임
- 판별망은 실제 학습 데이터와 생성망을 거쳐 만들어진 가짜 데이터를 이용하여 학습을 하며, 실제 데이터가 진짜인지 혹은 가짜인지 구별하는 역할을 함
- 생성망은 판별망을 속일 수 있을 정도로 진짜와 구별이 불가능할 수준의 가짜 데이터를 만들어내는 역할을 함
- 따라서 판별망은 판별을 잘 하는 방향으로, 생성망은 판별망을 잘 속이는 방향으로 계속 학습을 하다 보면 최종적으로 생성망은 진짜인지 가짜인지 구별이 불가능한 수준의 데이터를 만들어내고 판별망 역시 분별 능력이 점차 개선이 되는 것이 목표임
- 2장 가 절에서 나온 분류 모델 학습 같은 경우 대부분 비용함수를 정희하고 오차를 역전파 시키면서 변수를 갱신하는 방식을 사용함. 이는 특정 목적을 가진 모델 한 개를 학습시키는 것이기 때문임
- 하지만, 생성망과 판별망으로 이뤄진 적대적 신경망은 두 단계의 학습과정을 거치게 됨

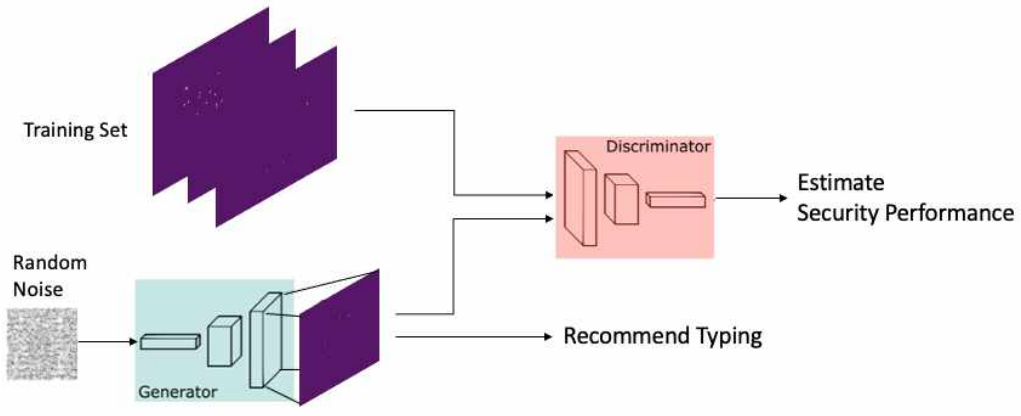


그림 10. 적대적 신경망 학습 다이어그램

- 첫 번째 단계에서는 생성망을 고정시키고 판별망을 학습 시킴. 판별망은 어느 것이 진짜이고 어느 것이 가짜인지 이미 알고 있기 때문에 기존에 사용하던 분류 모델의 학습 방법과 동일함
- 두 번째 단계는 판별망을 고정시키고 생성망을 학습 시킴. 생성망의 목적은 판별망을 속이는 것이기 때문에 판별망이 진짜로 착각할 수 있는 방향으로 학습을 수행함
- 위 두 과정을 반복하여 학습하다보면 생성망과 판별망이 발전을 거듭하여 평형상태에 도달하게 되면 학습이 종료됨

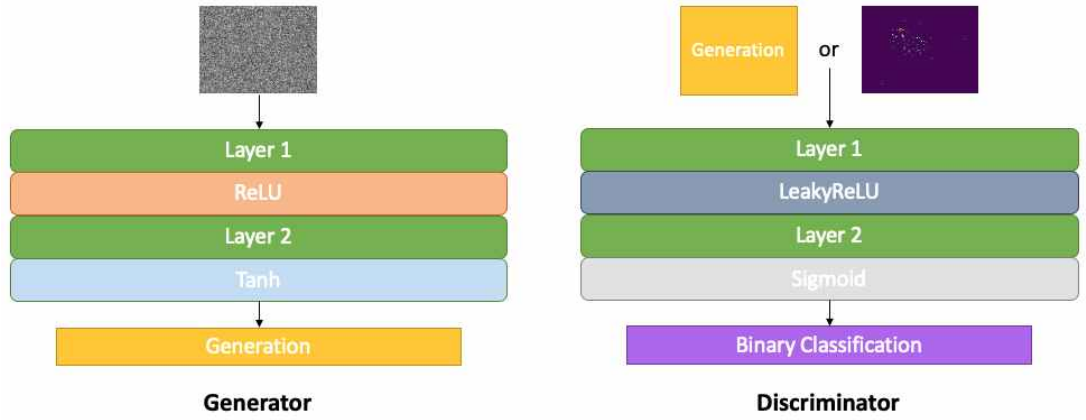


그림 11. 적대적 신경망 모델의 구조

- 그림 10과 11에서 보는 바와 같이 적대적 신경망의 입력으로는 무작위한 노이즈를 입력하여 생성망이 판별망을 속일 수 있는 데이터를 만들어 내는 것이 목표임
- 그림 11에서처럼 네트워크 구조를 구축하였음. 2장 가 절에서 다층 퍼셉트론 모델이 한 층의 깊이로도 충분한 성능을 보여주었기 때문에 기존 적대적 신경망 연구에서 적용한 모델보다는 얕은 네트워크를 구성함
- 하지만 적대적 신경망의 학습을 수행했으나 여러 문제점을 발견할 수 있었음

(나) 적대적 신경망의 문제점

- 첫 번째 문제점으로 생성망이 이미지를 잘 생성했다하더라도 해당 이미지를 단순히 시각화 테스트 이미지와의 거리 계산을 통해 유사도를 측정하는 것이 아닌 사용자가

사용했던 비밀번호를 생성하는 것이 목적이었음

- 하지만 시각화 이미지는 여러 복합적인 패턴을 순차적인 형태로 쌓아 만들었기 때문에 이를 다시 비밀번호로 복호화 하는 것이 불가능함
- 두 번째 문제점으로는 생성망이 생성해야 하는 이미지가 2장 가 절에서도 언급 했듯이 각 픽셀의 값이 주변 값들과 두드러지는 값을 가지고 있지 않음. 따라서 판별망과 서로 주고 받으며 학습이 돼야 하는데 판별망이 학습 초반부터 빠르게 수렴하여 생성망의 학습이 진행되지 않는다는 문제점으로 볼 수 있음

(다) 조건부 적대적 신경망 (conditional adversarial networks, CGAN[5])

- 기존의 적대적 신경망의 생성망은 무작위 노이즈를 입력으로 받아 학습 데이터의 분포를 반영한 실제와 비슷한 가짜 데이터를 출력함
- 무작위 노이즈는 내재 변수로도 불리며 직접적으로 관측되는 변수가 아니라 관측이 가능한 다른 변수들로부터 추론이 가능한 변수를 의미함
- 오토인코더 (autoencoder)에서처럼 입력 데이터를 받은 후 encoding 과정을 거쳐 압축된 정보를 만드는데 이것이 내재 변수임. 이를 다시 decoding 과정을 거쳐 데이터를 복원하게 되는데 decoding 과정만 본다면 생성망과 매우 유사함을 알 수 있음
- 무작위 노이즈라고 불리기도 하는 것은 사전에 정의가 불가능함을 의미에 둔 표현임
- 조건부 적대적 신경망은 데이터 생성에 관련된 조건을 부가하고 이것을 통해 데이터 생성을 도와주어 원하는 데이터를 생성할 수 있도록 하는 적대적 신경망임
- 조건부의 형태는 다양한 형태를 가질 수 있음
- Generator에 랜덤 노이즈와 함께 사용자 시각화 데이터를 합쳐서 사용해 노이즈만으로 예측하기 어려웠던 사용자의 비밀번호 사전 정보를 주입함

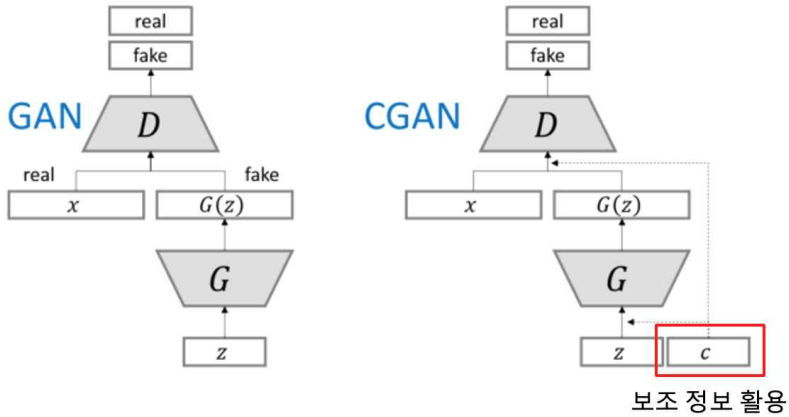


그림 12. GAN과 CGAN의 차이 비교 도식화

- 그림 6에서 적대적 신경망과 조건부 적대적 신경망의 차이를 확인할 수 있음
- 모델 설계자의 목적에 맞게 다양한 보조 정보를 활용할 수 있음

5) Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

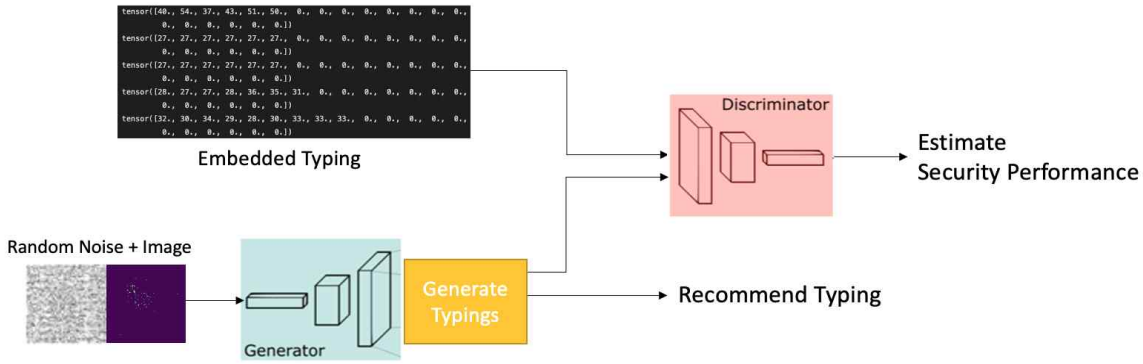


그림 13. 비밀번호 생성모델 학습 과정

- 그림 13에서는 본 절에서 제안하는 조건부 적대적 신경망의 학습 방법을 나타냄
- 앞서 적대적 신경망에서 접근했던 시각화 데이터 생성이 아닌 문자열 자체를 생성하는 조건부 적대적 신경망을 설계했음
- 무작위 노이즈만으로 직접 문자열을 생성하는 것이 아닌 특정 사용자의 시각화 이미지를 무작위 노이즈와 합쳐 직접적으로 사용자가 사용할만한 비밀번호를 생성하는 것임
- 이러한 방식을 통해 기존 이미지를 문자열로 decoding할 필요도 없으며 생성망은 사용자의 시각화 이미지의 입력 패턴을 고려하여 문자열을 생성할 수 있기 때문임
- 생성한 비밀번호 문자열 또는 사용자의 embedding된 실제 문자열을 discriminator의 입력으로 사용하여 서로 적대적 학습을 통해 generator가 최대한 사용자의 문자열과 유사한 비밀번호 문자열을 만드는 것이 목표임
- 그림 13과 같은 접근 방식을 통해 학습이 완료된 생성망이 비밀번호 문자열을 생성했을 때 사용자가 얼마나 비슷한 패턴으로 비밀번호 문자열을 생성해 사용하고 있는지 알 수 있음
- Discriminator 입력값과 generator 예측값은 0~100 사이로 변환된 비밀번호 문자열로 구성
- 사용하는 비밀번호 문자열의 최대 길이가 20이므로 이보다 작을 경우 “unk” 라벨을 사용하여 0으로 embedding과 decoding 실시함

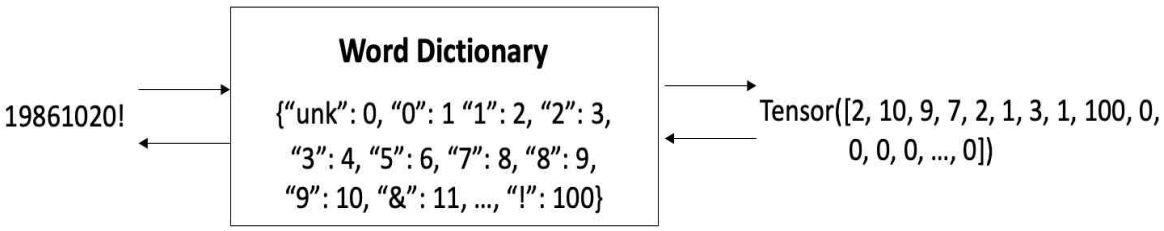


그림 14. 문자열 embedding, decoding 예시

- 그림 14와 같이 단어 사전을 생성해 모델의 입력으로 사용하기 위해 embedding 실시
- 사용된 단어 사전의 문자열은 시각화 데이터를 생성할 때 사용된 알파벳, 숫자, 보편적으로 사용되는 특수문자로 구성되어 있음

- embedding한 문자열로 조건부 적대적 신경망을 학습시키기 위한 모델의 구조는 아래 그림 15에서 확인할 수 있음

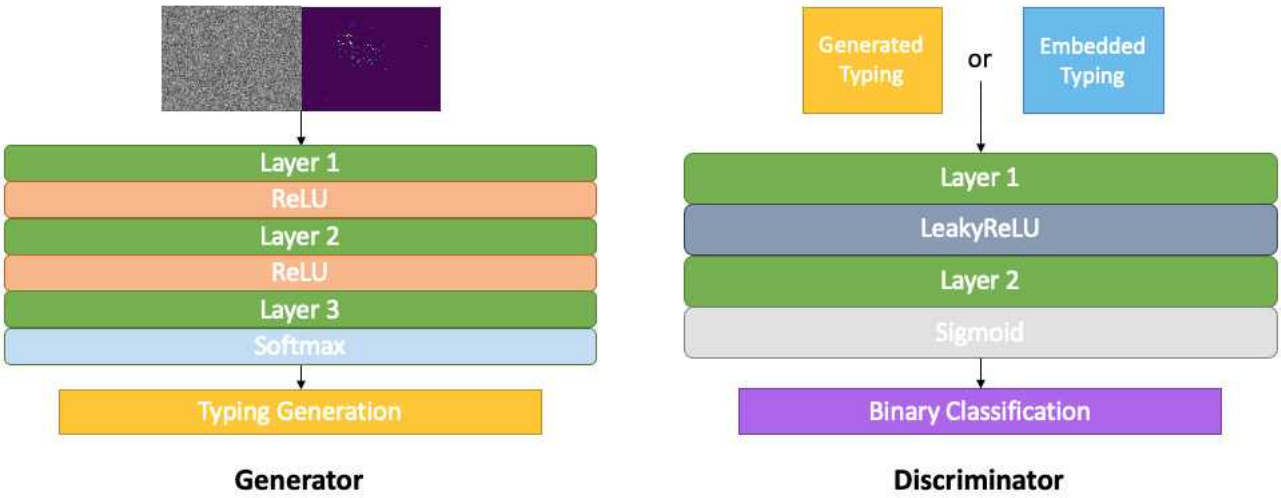


그림 15. 조건부 적대적 신경망 모델 (1)

- 그림 15에서 확인할 수 있듯이 조건부 적대적 신경망의 판별망 또한 기존 모델과 동일하게 이진 binary cross-entropy 비용함수를 사용해 학습을 수행함
- 무작위 노이즈와 사용자의 시각화 이미지를 생성망의 입력으로 사용해 최대한 사용자가 사용한 비밀번호와 유사하게 만드는 것이 목표임

User	Image	Original	Generated
10azzz		25021984 25021984 20180219 20180219	2502198745654 2512102197817953 251182919796887 252809579802513
10996		Qwe123 Qwerasdf123 qwerty123 qwe123	qwer123456 q2we2123r123123 qq1ert21q2 1q234qwe123eqw
1020		password1 password123 ppaasswwoorrdd passqwerty1	pasqwerd123 Pqw1e23ww111123 W12p2wowwd123 pqw12re34qwe1rd
10anya7		10012009 10012009 10012009 10012009	1000100009 1012340139 100000a12399 1212010019

표 2. 생성모델을 이진 cross-entropy loss만 사용하여 학습시킨 결과 예시

- 표 2에서는 적대적 신경망 모델 (1)을 통해 생성한 비밀번호와 실제로 사용한 비밀번호 간의 비교를 보여주고 있음
- 표 2에서 보면 생성망이 생성한 비밀번호의 앞부분을 보면 실제 비밀번호와 유사한 형태임을 확인할 수 있음
- 하지만 생성한 비밀번호와 실제 비밀번호의 길이를 보면 크게 차이나는 것이 보임
- 이는 생성망이 무작위 노이즈와 시각화 이미지만으로 길이를 맞추는 것이 어려운 것을 알 수 있음

- 일반적으로 CGAN에서는 비용함수로 이진 cross-entropy loss를 사용하지만 모델이 사용자의 문자열 길이를 정확히 판단하지 못하는 현상을 발견함
- 앞서 언급한 문제점을 해결하기 위해 그림 16처럼 생성망의 구조를 변경하고 비용함수를 추가함

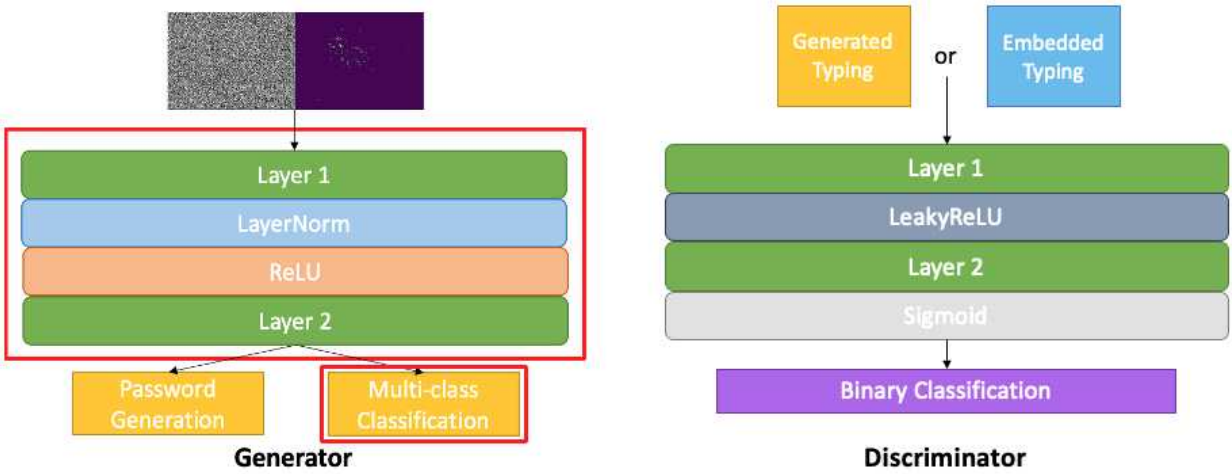


그림 16. 조건부 적대적 신경망 모델 (2)

- 다층 퍼셉트론의 분류 학습에서 발견 했던 내용처럼 조건부 적대적 신경망 모델 (1)에서도 다소 깊은 네트워크를 사용해 생성망이 올바르게 학습이 되지 않았던 것으로 해석함
- 따라서 생성망 네트워크의 깊이를 줄여서 작지만 세세한 정보 보다는 크고 넓은 정보를 추출할 수 있도록 구성함
- 또한, 생성하는 비밀번호의 길이를 사용자가 사용하던 비밀번호의 길이를 직접적으로 학습할 수 있도록 추가적인 비용함수를 적용함
- 앞선 2장 가 절에서 수행했던 사용자 분류 모델처럼 이번 생성모델에서는 사용자의 패스워드를 바탕으로 직접적으로도 분류 학습이 될 수 있도록 다중 분류 cross-entropy loss를 사용하여 문자열의 길이를 함께 학습하도록 구성함

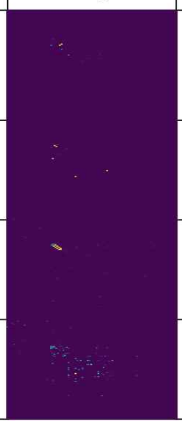
User	Image	Original	Generated
100.limites		fa656560 FA656560 fa6565 fa6565601	fa656560 fa656666 fa656560 fa656566
10010010010010010010010 0100		qwert123 qwe123 qwerty qwe123!	qwe123 qwer123 qwert123 qwer123
100101102		12345 1234554321 123456 123454321	123455 123456 1234554 12345321
10021978		cezelurreh cezelurreh198 cezelurreh1982 cezelurreh1982!	cezelurreh cezelurreh cezelurreh198 cezelurreh1982

표 3. 생성모델을 다중 분류 cross-entropy loss를 함께 사용한 결과 예시

- 표 3에서 생성망이 생성한 비밀번호와 사용자의 실제 비밀번호가 매우 유사한 것을 확인 할 수 있음
- 이는 다중 분류 cross-entropy 비용함수를 사용해 문자열의 길이를 함께 학습할 경우 사용자의 비밀번호 길이까지 맞추는 것을 확인할 수 있음
- 또한, 조금 더 얕은 네트워크를 사용해 이전 실험에서 살펴본 바와 같이 각 픽셀의 값이 크지 않은 시각화 데이터의 특성을 잘 추출할 수 있었음

(2) 생성 모델 성능 및 결과

- Top-k 정확도를 사용해 모델의 성능을 측정함
- 분류 모델에서 예측한 가장 가능성이 높은 k개의 클래스와 실제 클래스의 표준 정확도를 구하는 평가지표임
- 예를 들어 10 개 클래스 예측 모델의 클래스 별로 예측한 확률 값 중 k번째 안에 정답 클래스가 있다면 해당 예측은 맞는 예측으로 산정하여 대게 k로 1, 5, 10을 사용함
- 이러한 top-k accuracy를 차용해 생성모델이 k개의 비밀번호를 예측했을 때 사용자가 사용했던 비밀번호와 일치하는 개수로 생성모델의 성능을 평가함
- 데이터의 분할은 2장 가 절에서 적용한 바와 같이 학습 및 테스트셋은 8:2로 나누되 사용자가 실제로 사용했던 비밀번호는 학습과 테스트셋에 각각 분리되어 시각화 됐으며 서로 겹치지 않음
- 다중 분류 cross-entropy 비용함수의 라벨값으로는 테스트셋을 시각화하는데 사용된 비밀번호 중 랜덤하게 추출되어 사용됨
- 옵티마이저는 생성망, 판별망 모두 adam이 적용 됐으며 학습률은 동일하게 0.001로 적용함
- 조건부 적대적 신경망 (2)의 layer 또한 앞선 다층 퍼셉트론 분류 모델의 성능을 참고하여 합성곱 신경망이 아닌 fully-connected layer로 구성하였음
- 첫 번째 fully-connected layer는 은닉층 1024크기로 구성돼 있고 시각화 이미지와 함께 입력되는 무작위 노이즈는 100의 크기를 지니고 있음
- 그리고 마지막 fully-connected layer는 길이 20의 문자열의 각 101개의 문자를 예측해야 하기 때문에 크기 2020의 출력 크기를 지님

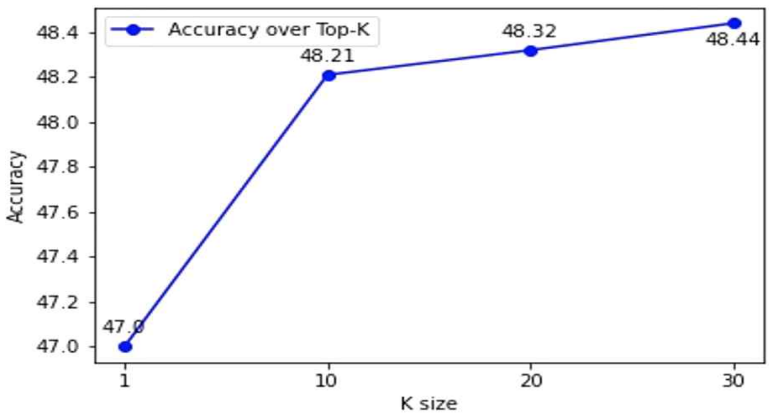


그림 17. 비밀번호 생성모델 Top-k 정확도

- 그림 10에서 보는 바와 같이 30개의 비밀번호를 생성망이 생성했을 때 평균 48.44%로 사용자의 실제 비밀번호를 예측하는 것을 확인할 수 있음
- 다만 통상적으로 k의 크기가 커질수록 정확도가 많이 상승해야 하는데 이러한 모습을 보여주지 않는 것으로 보아 생성모델이 사용자의 비밀번호 패턴을 기반으로 생성을 하기 때문인 것으로 해석함

User	Image	Password	Frequency
11shewolf		11shewolfq 1flowehs 1flowehs11 1kcuhC Chuck0 Chuck1 11shewolf1d	shewolf : 225 11shewolf : 92 shewolf1 : 62 shewol : 47 11shewolf1 : 43 11shewolf11shewolf : 26
100500tixover9009		tix100500123 tix100500A tix100500Q tix100500R Stix100500 Wtix100500 Xtix100500	tix100500 : 1485 1 : 218 2w : 2 at : 1

표 4. 비밀번호 생성모델 정성적 평가 결과

- 정성적 평가를 수행한 결과는 그림 11에서 확인할 수 있음
- 상단의 사용자는 생성망이 단 하나도 사용자의 실제 과거 비밀번호를 맞추지 못한 사용자이고 하단의 사용자는 모두 맞춘 사용자의 실제 이미지, 사용한 비밀번호, 사용한 비밀번호들의 빈도수를 보여주고 있음
- 두 사용자가 사용한 비밀번호 문자열을 비교해 보았을 때, 하단의 사용자는 frequency 열을 보면 한 개의 문자열을 압도적으로 중복되게 사용한 것을 확인할 수 있음
- 사용자가 사용한 비밀번호들의 패턴이 매우 유사한 형태를 할 경우, 생성모델의 예측이 정답과 일치한 경우가 많았으며 다양한 형태의 비밀번호를 생성해 해킹으로부터 방어가 필요함

(3) 알고리즘 코드 실행환경 및 사용방법

(가) 구동 하드웨어 및 소프트웨어 환경

- OS: Ubuntu 16.04
- CPU: Intel Xeon Gold 5220 (8 cores, 2.2 Ghz)
- GPU: Tesla V100-SXM2-32GB
- Python 3.6, Pytorch 1.7.1

(나) 코드 실행 방법

- 사용 데이터는 password/data/train, password/data/test 폴더의 이미지 사용
- 생성 작업 수행 코드 실행
- argument로 —train을 True 또는 False를 사용하여 미리 학습된 모델을 이용해 테스트

를 진행할지 새로 학습하여 테스트를 진행할지 선택할 수 있음

- 학습된 모델은 password/generation/ckpt 안에 저장되어 있음

```
$ cd password/generation
$ python generate.py --train True or False
```

다. 키 타이핑 사용 현황 분석

(1) 사용자별 타이핑 패턴 분석도 활용

- 개인정보 보안에 대한 중요성이 두드러지면서 패스워드에 대한 추측 공격에 노출되는 매우 취약한 문제들이 많이 발생함. 관련해서 기존 국내 사용자들의 패스워드 사용 현황 분석 [6]논문 리뷰를 통해 패스워드 관련 양적 연구를 시행함

(2) 설문조사 응답 모델 구축 및 사용자 특성에 기반한 패스워드 사용 분석

- 앞서 조사한 선행 연구들을 살펴보면 국내 웹 사이트들이 구성 청크 조합을 권고하였고, 패스워드 내 의미있는 구조들을 묶은 chunk의 using rate가 그림 18에와 같이 나타났으며 '이름'의 경우 약 46%의 사용률을 보이고 날짜(25%) 가 유의하게 빈번히 포함되었음 (χ2-test, p<0.001) 대부분 '날짜'는 숫자로 표현하는 반면, '이름'은 영문표기법, 한글, 머리글자, 기타외국어 등 더 복잡한 변환을 거친 방법으로 표현이 가능함
- 또한, '!', '@' 등 사용자들이 유의하게 선호하는 특수문자가 존재함을 그림 18 에서 볼 수 있고, 표현 가능한 의미 chunk들 중 자주 등장하는 '이름', '선호단어', '기타단어'들에 대해 어떠한 표현 방법들이 주로 사용되었는지 통계치를 보여줌

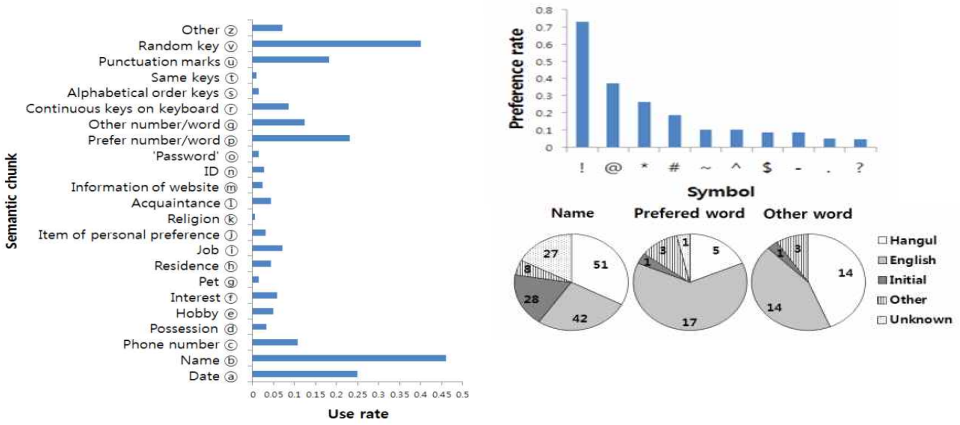


그림 18. 비밀번호 패턴조합 선행연구 통계데이터

- 설문 응답자들을 대상으로 예를 들면 '이영희' 라는 가상의 사용자의 인적 정보를 토대로 실제 응답자들이 이 가상의 사용자라 가정하면 쓸법한 패스워드들을 작성하는 시나리오를 그림 19와 같이 제작하였음

6) Kim, Seung-Yeon, and Kwon, Taekyoung. "ACase Study of Password Usage for Domestic Users." Journalof the Korea Institute of Information Security & Cryptology,vol. 26, no. 4, 한국정보보호학회, Aug.2016, pp. 961–972, doi:10.13089/JKIISC.2016.26.4.961.

그림 19. 문자 입력패턴 분석기술 설문조사 응답 예시 및 가상정보

- 총 20명의 가상정보를 토대로 실제 사용자 10명에 대해서 각각의 가상정보에 대해 5개씩의 비밀번호를 입력받아 데이터셋 확보함
  (Dataset : 20명(가상정보) * 10명(실제유저) * 5개 = 1000개)
- 확보한 데이터셋을 바탕으로 실제 설문대상 사용자들의 패스워드 선호도와 함께 semantic chunk usage rate를 만들어 사용자 10명중 5명의 유저를 뽑아 chunk frequency를 그림1과 같이 나타내었고, 전체 데이터셋에 대한 preference rate를 그림 20과 그림 21과 같이 표현함

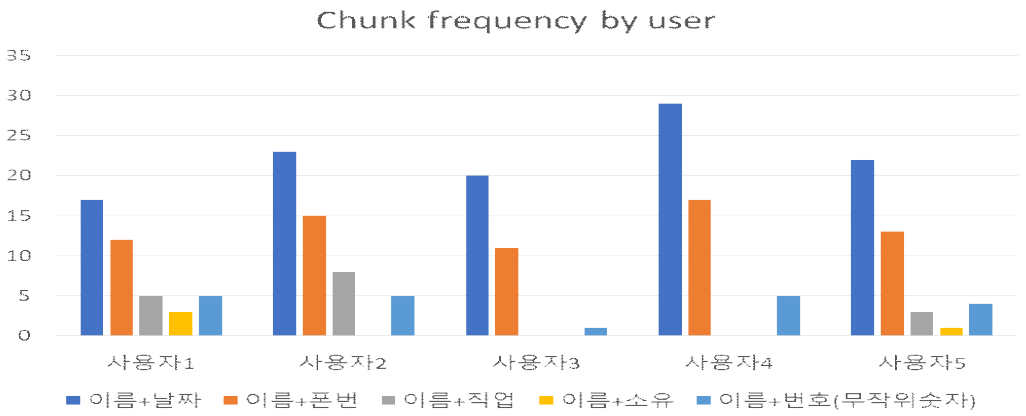


그림 20. chunk frequency by user chart data

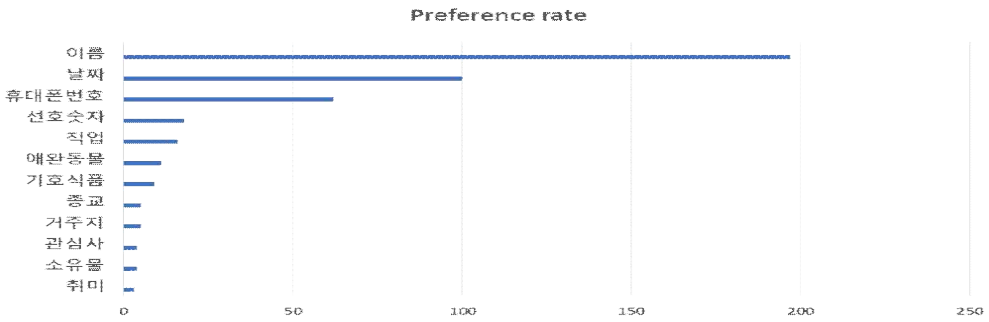


그림 21. preference rate chart

(3) 타이핑 패턴 기반 비밀번호 생성 알고리즘 구현

- 가상정보를 기반한 설문 응답 dataset을 바탕으로 새로운 비밀번호를 생성하여 실제 사용자가 쓸법한 패스워드 후보군 생성을 위한 알고리즘을 구현과정에서 데이터셋을 어떻게 정량화를 할지에 대한 방법론 구현이 필요함
- 기존 csv파일로 생성된 dataset을 매핑 및 전처리과정을 거치고 알고리즘 구현에 필요한 dataset을 수행시간 단축을 위한 5개의 가상정보를 택하여 변환과정을 통해 사용자가 쓸법한 비밀번호 그룹의 구성을 info_dict로 그림 22와 같이 구성함

```
info_dict = {
"name" : ["choi", "dudgml","cyh","choiyh", "C", "CYH", "Dudgml", "younghee", "yhyh"],
"date" : ["0310", "0921", "890921"],
"phone_num" : ["3265", "5485"],
"having" : ["5887"],
"sun_ho" : ["6"],
"gi_ho" : ["macarong","maca"],
"hobby" : ["cjdrPtks"],
"pet" : ["dusl", "Dusl"],
"religion" : [""],
"living" : ["4312"],
"job" : ["0516"],
"num" : [""],
"word" : ["@@", "!!", "##", "!", "@", "!@#"]
}
```

그림 22. visual studio code내 info_dict mapping 예시

- 이후 실제 사용자별로 이름, 날짜, 휴대폰 번호, 기호식품, 취미 등등 선호하는 패턴을 정량화하기 위해 추출한 5개의 정보에 대한 비밀번호 총 25개에 대해 가중치를 설정하기 위해 넘버링 및 전처리 과정을 그림 23과 같이 표현하여 수행함

	password	name	date	word	phone_nur	job	sun_ho	having	hobby	pet	living	gi_ho	religion
2	C3265YH548%	1		3	2								
3	C3265YJ5485	1			2								
4	dudgml0516	1				2							
5	dudgml051^	1		3		2							
6	Dudgml0516	1				2							
7	k2015sj0312	1				2							
8	k2015sj031@	1		3		2							
9	tjdwns0312	1				2							
10	tjdwns031@	1		3		2							
11	rla2015tjdwns031(	1		3		2							
12	dl96wldms0620	1	2										
13	dl96wldms062)	1	2	3									
14	Dl96wldms0620	1	2										
15	Dl96wldms062)	1	2	3									
16	Easysilver062)	1	2	3									
17	Tngus0921!	1	2	3									
18	tngus90921	1	2										
19	tngus9092!	1	2	3									
20	Tngus90921	1	2										
21	Tngus9092!	1	2	3									
22	wjdtlr1029	1	2										
23	Wjdtlr1029	1	2										
24	wjdtlr102(	1	2	3									
25	Wjdtlr102(	1	2	3									
26	dlwjdtlr1029	1	2										

그림 23. 가중치 설정을 위한 전처리과정 csv 예시

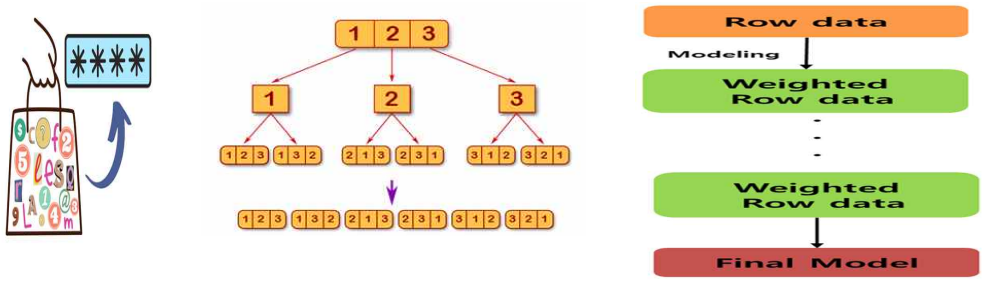


그림 23. 도식화 과정

- 다음 그림 23의 도식화 과정과 같은 가중치를 바탕으로 실제 사용자들의 비밀번호 조합들의 종류와 개수를 계산하여 각 비밀번호 조합들의 빈도수를 파악하고 빈도를 통해 weight를 구하게 됨(weight_dict)
- weight가 없는 경우는 전부 동일한 가중치로 without_weight로 생성
- python 환경에서 pandas 라이브러리를 활용하여 dictionary_function을 활용해 mapping 과정을 거친 mapped_data를 list로 return을 함
- 이후 info_dict의 column name들을 대상으로 순열 조합을 통해 logic 생성
  (e.g. 기호+식품, 기호+식품+동물, 동물+기호+식품 ...)
- 새로운 패스워드를 생성할 때 만들어진 chunk 조합들을 아래 그림 24과 같이 결과들이 나타내고 해당 chunk 조합들의 weight 가중치를 계산하여 수치로 표현하는 과정을 나타낼수 있음

그림 24. 패스워드 조합 리스트 및 가중치 계산 결과

(4) Data mapping sequence - i

- 사용자별 타이핑 패턴분석 방법에서 실제 사용자의 설문응답 dataset 중 ¼ 정도의 무작위추출 시행을 하였고 User1: 100개(20*5) , User2-5 : 25개(5*5) 비밀번호를 활용하여 알고리즘에 적용하여 실제 사용자가 사용한 패스워드 chunk들로만 info_dict를 구성함
- info_dict의 input을 통해 총 50번의 알고리즘 수행을 반복해 몇번째 시도만에 동일한 비밀번호를 찾는지에 대해 평균을 내어 가중치를 적용하였을때, 적용하지 않았을때, 조합자체 랜덤일경우의 세 가지 경우에 대해 시행을 함
- 이를 통해 특정 사용자에 대한 타이핑 패턴의 중요성 확인과 일반적 사용자들에 대한 타이핑 패턴의 중요성 비교를 하고 대조군으로 완전 랜덤 조합 시행으로 사용자별

타이핑 패턴 분석에 의의를 둠

(5) Data mapping sequence - ii

- 실험 i)과 달리 알고리즘 logic은 그대로 적용하면서 info_dict의 input data를 실제 사용자가 작성한 패스워드가 아닌 가상정보가 주어진다면 해당 사용자가 쓸법한 패스워드의 후보군을 임의로 작성하여 chunk 작업을 통해 input을 생성함.
- 이후 실제 패스워드와 일치할 때까지 무작위 조합을 통해 실험 i과 마찬가지 동일하게 실제 패스워드와 일치할 때까지 반복시행하며 생성된 비밀번호들의 조합들도 같이 확인하는 형태의 로직을 그림 25와 같이 구성

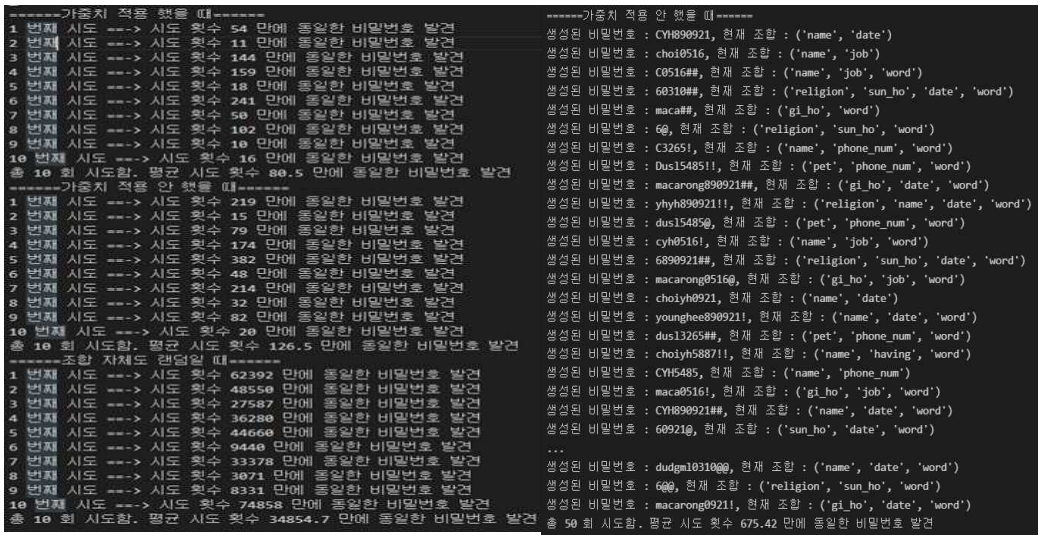


그림 25. 패스워드 조합 리스트 및 가중치 계산 결과

(6) 생성 모델 성능 및 결과

50회시도 평균	user1	user2	user3	user4	user5	user1	user2	user3	user4	user5
가중치적용	190회	114회	9회	72회	OOM	414회	OOM	299회	28회	64회
가중치적용x	403회	141회	13회	114회	OOM	675회	OOM	1777회	37회	98회
랜덤 조합	OOM	23127회	363회	36104회	OOM	OOM	OOM	OOM	OOM	OOM

표 5. 유저 별 매칭을 통한 생성 실험 결과

- 패스워드 후보군에 실제 설문응답한 비밀번호를 바탕으로 매핑을 진행한 data mapping sequence i 표5 좌측처럼 보일수 있으며 user5의 경우 데이터 전처리 과정 이후 조합의 경우가 무한에 가까워 OOM(Out of Memory)를 보임을 알 수 있음
- 이와 달리 가상정보를 보았을 때 쓸법한 비밀번호의 조합을 매핑한 data mapping sequence ii 표5 우측을 통해 보일 수 있으며 user2의 경우 쓸법한 비밀번호 조합의 구성으로 매칭을 해보았지만 다양한 구성의 조합으로 보안성이 높아 수많은 trial에도 불구하고 해당 비밀번호를 찾지 못함을 알 수 있음
- 완전 랜덤 조합의 경우는 수많은 trial이 수반되어도 기존의 password를 찾지 못함을 나타내며 특정사용자에 대한 타이핑 패턴의 중요성과 일반적 사용자들에 대한 타이핑 패턴의 중요성의 비교를 통해 확인할 수 있음

(7) 알고리즘 코드 실행환경 및 사용방법

(가) 구동 하드웨어 및 소프트웨어 환경

- OS: Windows 11
- CPU: 11th Gen Intel Core i7 2.80GHz
- Memory: 16GB RAM
- Python 3.913, Jupyter notebook 9.1303

(나) 코드 실행 방법

import pandas as pd # 엑셀 csv파일로 정리된 dataframe을 읽기 위해선 pandas 라이브러리가 필요함.

raw_data = pd.read_csv(“mapped_data.csv”) # csv파일로 저장한 mapped_data를 불러옴

def get_pw_comi_list(): # mapped_data를 가지고 각 유저별 비밀번호 조합을 통해 list로 return

def get_weight_dict(): # 각 유저별 비밀번호의 조합들의 종류, 개수를 파악해 빈도를 파악

def get_permute(_dict): # info_dict의 column name을 가지고 순열을 제작. e.g. 이름+기호

permute += list(itertools.permutations(li, count)): #python 내 itertools 패키지 library를 활용해 iterable 객체의 원소들로 구성되어 가능한 모든 순열을 구하는 함수를

def get_result(cur_count, avg, weight_dict): # 현재 진행된 시도 횟수와 평균 및 dictionary list에 저장된 가중치 weight을 토대로 result를 내며 이 과정에서는 random 난수 함수를 통하여 combination 조합을 살펴봄

def get_result_random(cur_count, avg, permute) # 현재 진행된 시도횟수, 횟수의 평균, 순열조합을 활용한 data mapping sequence I의 완전랜덤조합을 구하기 위하여 정의를 함

pw_combi_list = get_pw_combi_list()
weight_dict, without_weight = get_weight_dict() # 이미 지정한 def함수들로부터 정의를 통해 마지막 print 출력을 통해 새로운 비밀번호 생성과 함께 매칭과정을 거치게 됨

라. 사용자 특성 기반 Application 연구

(1) 사용자 특성(게시글 등)에 기반 분류 기법 연구

- 최근에는 특정 데이터에 대한 사용자들의 평균적인 반응 및 응답을 예측하는 수준을 넘어 개별 사용자의 특성을 분석하는 연구가 수행되기 시작하는 단계임. 하나의 예로 이미지에 대한 일반적인 설명 문장을 생성하는 image captioning 연구의 경우, 최근에는 개별 사용자마다 같은 이미지에 대해서도 서로 다른 내용과 스타일의 설명을 할 수 있다는 보다 실질적인 가정아래 *personalized* image captioning 연구가 수행[7)]
- 본 연구에서는 이러한 최신 연구 동향을 바탕으로 사용자가 기존에 작성하던 해쉬태그 패턴을 분석 및 추출하여, 이로부터 새로운 이미지가 주어졌을 때, 사용자별 서로 다른 패턴의 해쉬태그 (개인화된 해쉬태그)를 생성함. 그다음, 이렇게 생성된 개인화된 해쉬태그를 바탕으로 주어진 이미지에 대해 어떤 카테고리의 게시글을 작성하였는 지 예측하는 모델 설계 연구를 수행함.
- 또한, 해당 연구에 적합한 공개 데이터셋의 부재 문제를 해결하기 위해 기존의 단일 유저별 이미지 및 게시글 쌍 데이터셋인 YFCC100M을 기반으로 게시글의 주요 카테고리별 게시글과 유저 정보를 추출하여 본 연구에 적합한 데이터셋을 새로이 구축
- 실험을 통해 제안하는 모델이 사용자의 해쉬태그 작성 특성을 잘 학습하여 주어진 이미지 종류에 적합한 해쉬태그를 생성해내고, 사용자 특성 기반으로 생성된 해쉬태그가 이미지 카테고리 종류에 알맞은 해쉬태그인지를 정량적 평가를 통해 검증함.
- 특히, 사용자의 성향, 관심에 따라 동일한 이미지에 대해서도 세부적으로는 다른 카테고리에 해당하는 해쉬태그를 생성할 수도 있음. 예를 들어, 공연 이미지가 주어졌을 때, 어떤 사용자는 콘서트와 관련된 해쉬태그를 생성하고, 어떤 사용자는 음악과 관련된 해쉬태그를 생성할 수도 있음. 본 연구에서는 실험을 통해 제안하는 모델이 사용자가 기존에 작성했던 해쉬태그의 패턴을 학습하여 같은 이미지 데이터에 대해서도 각자의 성향 특성에 따라 다른 종류의 해쉬태그를 생성함과 생성된 해쉬태그가 사용자별 관심 카테고리에 다르게 분류될 수 있음을 시각적 결과로 보임

(2) YFCC100M-20Class 데이터셋 구축

- 원시 데이터셋인 YFCC100M[8)]은 2004년부터 2014년까지 Flickr에 업로드 된 multi-media 들을 모은 dataset으로, 약 57만 명의 사용자의 9,920만 장의 사진과 80만 개의 비디오 제목, 설명, 카메라 유형, 사용법 등의 해당 메타데이터를 포함
- YFCC100M 데이터셋에서는 사용자가 작성한 태그를 해시태그(Hashtag)로 간주하였고 image captioning을 하기 위해 원시 데이터셋을 특정한 기준에 따라 필터링(Filtering)하여 5,495명의 434,936개의 게시글을 확보하였고 그 결과는 그림 26과 같음
- 하지만 유저가 직접 작성한 태그를 내용을 토대로 데이터셋을 구축하였기에 특정한 기준이 없이 너무나 자유로운 형태로 데이터셋이 구성되어있기에 데이터셋을 분류하기에 어려움이 존재함

---

7) C. C. Park, B. Kim, and G. Kim, “Towards Personalized Image Captioning via Multimodal Memory Networks,” IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI), vol. 41, no. 4, pp. 999 - 1012, Apr. 2019.

8) B. Thomee et al., “YFCC100M: The New Data in Multimedia Research,” Communications of the ACM, vol. 59, no. 2, pp. 64 - 73, Feb. 2016.

- 본 연구에서는 새롭게 구축한 YFCC100M-20Class 데이터셋은 YFCC100M 전체 데이터셋의 태그 빈도를 확인하여 사용자가 주로 사용하는 해시태그를 파악하고 상위 20개의 태그들을 선별하여 라벨링(Labeling)을 진행함. 태그의 종류는 표 6과 같음

그림 26. 기존 연구의 데이터셋 (YFCC100M 필터링 적용 후의 결과)

- 하나의 게시글은 여러 개의 Class를 가질 수 있으며 만약 위에 해당하는 태그가 존재하지 않을 시 해당 게시글을 제외하는 것으로 필터링함. 위 과정을 거쳐 총 4,048명의 137,028개의 게시글을 확보하였고 이 데이터셋을 YFCC100M-20Class로 명명하고 그 내용은 그림 26와 같음

Flicker tag	travel	wedding	flower	art	music
태그 수	1,221,148	734,438	907,773	902,043	826,692
Flicker tag	party	nature	beach	city	tree
태그 수	669,065	872,029	768,752	701,826	697,009
Flicker tag	vacation	park	people	water	architecture
태그 수	694,523	686,458	641,571	640,259,	616,299
Flicker tag	car	festival	concert	summer	sport
태그 수	610,114	609,638	605,163	601,816	564,703

표 6. YFCC100M에서 가장 자주 사용되는 20개의 태그

- 마지막으로 데이터에 대하여 전처리과정을 실시함. YFCC100M-20Class 데이터셋에서 가장 빈번히 사용되는 40,000개의 단어를 선택하여 단어 사전(Dictionary)을 구성함. 딕셔너리를 구성하기 전에 태그에서 URL, 특수문자 및 유니코드 기호들을 제거하였고, 대문자로 작성된 태그가 존재하였다면 소문자로 변경함. 이후 사용자의 게시글에서 앞서만든 단어 사전에 포함되지 않은 나머지 태그들을 제거함
- 본 연구팀이 YFCC-100M 를 연구 목적에 맞게 수정 및 구축한 YFCC100M-20Class 데이터셋을 github (https://github.com/hanu14/Classifyer_Irislab)에 공개

그림 27. 새롭게 구축한 데이터셋, YFCC100M-20Class

(3) 게시글 특성의 패턴을 학습하여 게시글을 분류하는 모델 개요

- 본 연구에서는 사용자가 특정 이미지를 보고 작성한 게시글 (해쉬태그)를 분석하여 이를 바탕으로 어떤 카테고리의 해쉬태그를 작성했는 지 예측하는 개인화된 게시글 카테고리 예측 연구를 수행하고자 함
- 이를 위해 제안하는 모델은 크게 이미지 및 사용자별 해쉬태그 특성을 추출하는 사용자 특징 추출 모듈과 추출된 사용자별 특징을 바탕으로 주어진 이미지에 대해 사용자가 작성한 해쉬태그가 어떤 카테고리의 내용인지 분류하는 카테고리 분류 모듈로 구성됨
- 먼저, 주어진 이미지에 대해 사용자가 작성한 개인화된 해쉬태그 정보를 추출하기 위해 본 연구팀은 인스타그램 게시글을 분석하여 사용자별 맞춤형 해쉬태그 및 게시글을 생성하는 Context Sequence Memory Network (CSMN) 모델을 본 연구의 백본 네트워크로 사용함 (https://github.com/cesc-park/attend2u). 해당 모델을 바탕으로 YFCC100M-20Class 데이터셋 내 사용자별 이미지 및 그에 대응되는 개인별 해쉬태그의 특성을 학습함으로써, 주어진 이미지에 대해 사용자별 유의미한 특징 벡터(Feature vector)를 추출함
- 그 다음, 사용자별 해쉬테그 카테고리 분류 모듈에서는 앞선 백본 네트워크로부터 추출된 사용자별 해쉬태그 및 게시글 특징 벡터를 입력으로 받아서 사용자가 작성한 해쉬태그가 본 연구팀이 구축한 YFCC100M-20Class 내 20개 카테고리 중 어떤 카테고리에 속하는 지를 예측 및 분류함
- 본 연구에서 제안하는 네트워크 구조는 그림 3과 같음 (자세한 설명은 “o 신경망 모델” 파트 참고)
- 백본 네크워크에서 얻은 특징 벡터를 토대로 게시글을 분류하는 모델을 구축하고 이를 코드 형태로 공개 (https://github.com/hanu14/Classifyer_Irislab)
- 사용자는 해당 코드를 내려받아 데이터셋을 학습하고, 게시글의 특징에 기반한 카테고리 분류 예측의 정확도를 검출해볼 수 있음
- 또한, 공개된 github 저장소에는 초기 데이터셋을 가공한 새로운 데이터셋과 기학습된 초기 딥러닝 모델을 제공함

(4) 신경망 모델

- 사용자 특징 추출 모듈은 메모리 네트워크(Memory Network)를 백본 네트워크로 활용하

는 딥러닝 네트워크을 사용하여 구축되어짐

- 일반적으로 글의 내용을 분석하는 등의 연속적인 입력이 존재하는 경우 연속적인 데이터의 순서 정보를 기억할 수 있는 순환 신경망(Recurrent Neural Network, RNN)을 주로 사용함. 하지만 순환 신경망과 그것의 변형들은 가중치를 업데이트 하는 방법으로 Backpropagation Through Time (BPTT) 알고리즘을 사용하는데 데이터의 시퀀스(Sequence)가 길어질 경우 그레이던트 베니싱(Gradient Vanishing)이 발생하면서 장기 의존성(Long-term dependency)을 포착하는 것에 어려움을 겪음. 추가로 SNS의 게시글은 다양한 주제가 존재하며 그 주제에 대하여 사용자들의 게시글 작성 스타일 또한 다양하기에 사용자나 이미지의 메타 데이터(Meta data)에 대한 사전 지식(Prior knowledge)을 활용하는 것이 용이하지만 순환 신경망의 경우에는 이것이 불가능
- 특징 추출 모듈은 메모리 네트워크를 사용함으로써 앞서 언급한 단점을 극복함. 첫 번째로 메모리에 네트워크의 출력을 순차적으로 저장하여 모든 단어 정보를 메모리 안에 명시적으로 유지하기에 그레이던트 베니싱(Gradient Vanishing) 문제를 해결할 수 있으며, 두 번째로 이러한 정보를 학습의 사전 지식으로 사용이 가능함
- 또한, 메모리를 사용함으로써 이전에 생성된 단어와 사용자가 주로 사용하는 단어들과의 조합을 선택적으로 사용할 수 있음
- 하지만 기본적인 메모리 네트워크는 메모리 사이의 상관관계를 나타낼 수 없기 때문에 성능이 낮으며, 이를 위해 합성 곱 신경망(Convolution Neural Network, CNN)을 메모리 네트워크에 적용함으로써 성능을 고도화시킴

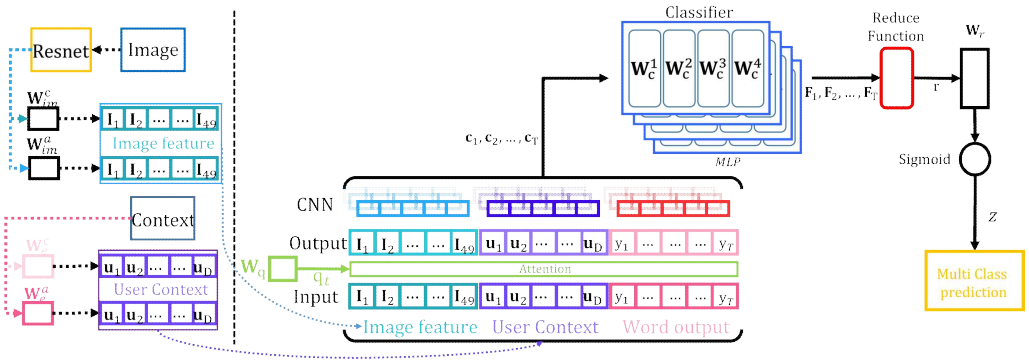


그림 28. 제안하는 사용자별 게시글 생성 및 카테고리 분류 네트워크 구조도

- 사용자 특징 추출 모듈의 학습 과정 및 방법은 아래에 기술함
- 앞선 과정을 거쳐 특징 추출 모듈이 출력한 특징 벡터에는 이미지, 해시태그, 사용자의 특성에 대한 정보가 내재되어 있음. 하지만 그 특징 벡터만으로는 어떠한 정보가 카테고리를 표현하는데 중요한 역할을 하는지 알 수 없으며 어떠한 값끼리 조합을 해야 최종적인 결과가 나타나는지 알 수 없음. 따라서 획득한 특징벡터를 카테고리 분류 모듈을 통하여 최종적으로 카테고리를 분류함
- 카테고리 분류 모듈은 암시적 신경 표현 모델(Implicit Neural representation)을 바탕으로 구축됨. 암시적 신경 표현은 어떠한 것인지 알 수 없는 정보를 가지고 있는 데이터가 존재할 때 그 데이터가 갖고있는 정보를 딥 네트워크를 통해 표현하는 방식
- 카테고리 분류 모듈은 다중 레이어 퍼셉트론(Multi-Layer Perceptron, MLP) 네트워크를

사용하여 특징 추출 모듈에서 얻은 특징 벡터를 특정 카테고리로 가장 잘 표현하는 신경망을 구성함. 다중 레이어 퍼셉트론의 기본적인 구조는 그림 4와 같으며 그림에서 볼 수 있듯이 비교적 간단한 네트워크 구조를 가지면서 동시에 데이터에 내재된 정보를 파악하는데 뛰어난 성능을 보이기에 최근 다양한 인공지능 연구분야에서 다중 레이어 퍼셉트론 네트워크를 많이 활용

- 카테고리 분류 모듈의 입력으로 들어온 특징 벡터는 레이어를 통과할 때마다 서로 조합을 하며 다른 형태의 특징 벡터로 변화함. 마지막 출력 레이어를 지나면서 카테고리를 예측하여 확률의 형태로 표현하고 실제의 표현과 예측된 표현 간의 차이를 통하여 네트워크의 학습을 진행. 해당 과정을 계속 반복하면서 네트워크는 입력된 특징 벡터를 확률의 형태로 카테고리를 가장 잘 표현하는 모델로 업데이트됨

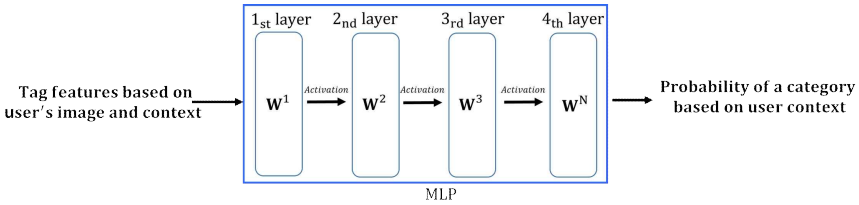


그림 29. 다중 레이어 퍼셉트론

(5) 알고리즘 학습

- 본 연구에서 제안하는 알고리즘의 학습 방법은 특징 추출 모듈을 우선적으로 학습하여 게시글의 이미지 및 사용자별 해쉬태그 특성에 대한 특징 벡터를 생성하고 특징 벡터를 통하여 카테고리 분류 모듈을 별도로 학습하는 것임
- 각 네트워크는 별도로 학습되기에 네크워크별 파라미터 간 간섭을 하지 않음

(6) 사용자 특징 추출 모듈

- 본 연구에서 CSMN 모델을 백본 네트워크로 하여 구성한 사용자 특징 추출 모듈은 세 개의 메모리 슬롯으로 구성된 메모리 네트워크를 사용함. 각 메모리에는 게시글별 이미지, 사용자가 주로 사용하는 태그 와 네트워크를 통해 생성된 단어가 저장
- 게시글의 이미지는 레지듀얼 네트워크(Residual Network)를 사용하여 이미지를 입력 특징 벡터로 만들며 그 정보를 저장함
- 사용자별 주 사용 태그는 TF-IDF Score를 가중치로 사용하여 사용자별로 전체 게시물에서 가장 자주 사용되는 D개의 단어를 선택하여 그것을 원 핫 벡터(One-hot vector)로 만들고 이것을 입력 특징 벡터로 만들며 그 정보를 저장함. 본 연구에서는 해당 정보를 사용자의 컨텍스트 정보 (user context)라고 부름
- 또한, 이번 게시글에 한해서 네트워크를 통해 이전에 예측된 단어를 원 핫 벡터로 만들어 이것을 입력 특징 벡터로 삼아 메모리에 저장하고 이러한 세 개의 메모리 슬롯을 결합하여(Concatenate) 입력 메모리를 구성함
- 입력 메모리는 이전에 예측된 단어들과의 관계를 가중치로 사용하는 어텐션 모델을 통해 출력 메모리를 구성하고 출력 메모리는 다시 3개의 메모리 슬롯으로 분해됨
- 각각의 메모리는 합성곱 신경망을 통과하며 메모리에 저장된 정보와의 관계를 고려한 3종류의 특징 벡터(이미지, 주사용 단어, 예측된 게시글에 대한 정보가 내재되어 있음)를

출력함

- 각각의 출력 특징벡터는 결합되고(Concatenate) 더해지며(Addition) 네트워크의 최종 출력 특징 벡터를 만들며 그 것을 통해 새로운 단어를 예측하고 이 단어를 메모리에 업데이트 하면서 동시에 네트워크 파라미터의 가중치도 업데이트함
- 사용자 특징 추출 모듈이 학습되는 동안에는 카테고리 분류 모듈의 파라미터는 동결(Freeze)시켜 별도로 학습이 진행됨

(7) 카테고리 분류 모듈

- 카테고리 분류 모듈은 사용자 특징 추출 모듈을 통해서 얻은 특징 벡터를 분류 네트워크의 입력으로 사용함
- 특징 벡터는 게시글의 특성에 대한 정보가 혼합되어 있지만 단순히 그 정보만으로는 카테고리를 표현하기에 어려움이 존재함
- 분류 모듈은 4개 레이어로 구성된 다중 레이어 퍼셉트론을 사용하여 이러한 특징 벡터의 패턴을 학습하여 새로운 형태의 특징벡터로 만듦
- 하나의 게시글이 T개의 태그를 가지고 있을 때 전술한 특징 벡터 또한 T개가 존재
- 하나의 벡터는 동일한 크기를 가지고 있어 MLP의 학습에는 영향을 주지않음. 하지만 게시글마다 전체 벡터의 개수는 다르기에 최종 레이어를 통과하여 고정된 형태의 확률로 표현하기 어려움
- 따라서, 이 특징 벡터를 결합(Concatenate)한 후에 정적 차원 축소 알고리즘을 이용하여 동적인 차원을 가진 특징 벡터를 정적인 차원으로 축소시키는 과정이 필요 (자세한 내용은 “o 정적 차원 축소 알고리즘” 파트 참고)
- 이후에 정적 차원으로 축소된 특징벡터는 최종 레이어를 통과하여 각 카테고리별 0~1 사이의 확률로 표현함
- 실제 카테고리와 확률로 예측된 카테고리 사이의 이진 교차 엔트로피(Binary Cross-Entropy, BCE)를 구하여 서로를 비교하고 이것을 네트워크의 손실 함수로 사용하여 네트워크의 학습을 수행
- 카테고리 분류 모듈이 학습되는 동안에는 특징 추출 모듈의 파라미터를 동결시켜 분류 네트워크의 입력으로 들어오는 특징벡터값의 영향을 주지않음

(8) 정적 차원 축소 알고리즘

- 본 연구에서 사용한 정적 차원 축소 알고리즘은 3종류가 존재하며 그 종류는 그림 5를 통해 확인할 수 있음. 그림 30에서 열 방향은 특징 벡터이며, 특징 벡터가 행 방향으로 결합되어 하나의 행렬(Matrix)을 구성한다고 가정함
- 첫 번째 방법은 행렬의 최댓값만을 획득하는 Reduce_Max 알고리즘. Reduce_Max 알고리즘은 행렬이 들어오면 임의로 지정한 방향으로 최댓값만을 획득하여 행렬을 축소시켜 벡터로 만듦. 그림 30의 (a)에서 볼 수 있듯이, 색이 특징 벡터의 값이며, 채도가 진할수록 큰 값을 가지고 있다고 가정함. Reduce_Max 알고리즘을 적용한 축소된 특징 벡터 r은 행방향으로 가장 큰 값(채도가 높은 색)만을 저장함
- 두 번째 방법은 행렬의 값의 평균을 획득하는 Reduce_Mean 알고리즘. Reduce_Mean 알

고리즘은 행렬이 들어오면 임의로 지정한 방향의 값의 평균을 획득하여 행렬을 축소시켜 벡터로 만듦. 그림 30의 (b)에서 볼 수 있듯이, 색이 특징 벡터의 값이라고 가정함. Reduce_Mean 알고리즘을 적용한 축소된 특징 벡터 r은 행방향으로 평균(색의 혼합)만을 저장함

- 세 번째 방법은 임의로 행렬을 자르는 Stack&Slice. 행렬이 들어오면 행렬을 임의의 크기로 잘라서 정적인 차원으로 축소시키고 나머지 특징 벡터를 사용하지 않음. 그림 30의 (c)에서 볼 수 있듯이, 다른 그림들과 다르게 색이 특징 벡터라고 가정함. 따라서 하나의 특징벡터는 같은 색을 가짐. Stack&Slice 알고리즘을 적용한 축소된 특징 벡터 r은 행렬을 열방향으로는 전체, 행방향으로는 사용할 만큼 잘라내고 잘라낸 행렬을 열방향으로 쌓아 벡터로 만듦

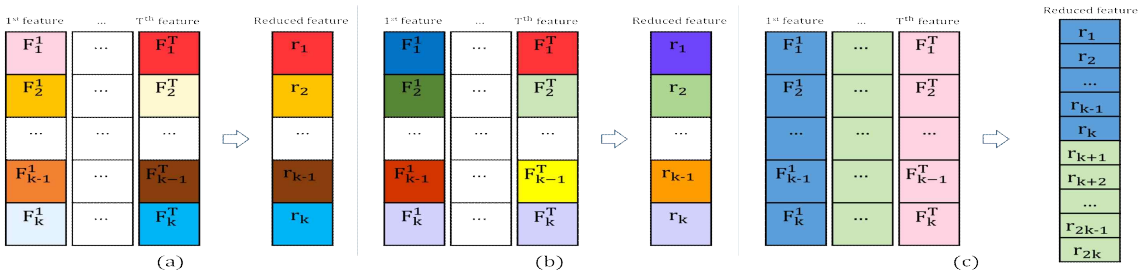


그림 30. 정적 차원 축소 알고리즘

(a)는 각각의 특징벡터를 비교했을 때 최대 출력값만 획득하는 Reduce_Max, (b)는 각 특징벡터의 평균값만을 획득하는 Reduce_Mean, (c)는 특징 벡터의 일부만을 선택하여 잘라 획득하는 Stack&Slice

(9) 실험 환경

- 본 연구에서는 사용자 특징 추출 모듈이 게시글의 내용을 충분히 학습할 수 있도록 배치의 크기는 200으로 총 20 epochs를 학습시킨 후에 카테고리 분류 모듈은 배치 크기 200, Epochs는 5 ~ 20으로 5 epochs의 차이를 두어 학습하였고 그 결과를 비교
- 각각의 네트워크는 Adam optimizer를 사용하여 최적화하였으며 초기 학습률은 0.001로 설정하고 5 epochs마다 학습률에 0.8을 곱하여 축소함
- 카테고리 분류 모듈의 정적 차원 축소 알고리즘으로는 3개의 특징 벡터를 선택한 Stack&Slice를 사용함

Epoch	#Top-predictions	Accuracy	Solver	Batch	Learning rate	Decay epoch	Decay rate
5	1	66.5%	Adam	200	0.001	5	0.8
10	1	68%	Adam	200	0.001	5	0.8
15	1	67.1%	Adam	200	0.001	5	0.8
20	1	67.6%	Adam	200	0.001	5	0.8
5	2	81.8%	Adam	200	0.001	5	0.8
10	2	82.6%	Adam	200	0.001	5	0.8
15	2	82.4%	Adam	200	0.001	5	0.8
20	2	82.4%	Adam	200	0.001	5	0.8

표 7. 분류 네트워크의 메인 카테고리 예측 성능

Epoch	Accuracy	Precision	Recall	F1-score
5	0.956	0.694	0.553	0.616
10	0.958	0.699	0.595	0.643
15	0.957	0.692	0.581	0.632
20	0.957	0.689	0.600	0.641

표 8. 분류 네트워크의 Confusion matrix 평가 결과

(10) 실험 결과

- 표 7는 사용자별 생성된 해쉬테그를 기반의 카테고리 예측 확률이 가장 큰 것의 갯수에 따라서 그 정확도를 나타냄. 카테고리의 예측 확률이 가장 높다는 말은 게시글을 가장 잘 표현하는 카테고리를 예측한 것임 즉, 게시글의 메인 카테고리를 예측한 것이 얼마나 정확히 예측되었는가를 판단한 결과임
- #Top prediction이 예측한 메인 카테고리의 개수이며 1의 경우는 카테고리 예측 확률이 가장 큰 것이 실제 카테고리인지 확인하여 맞다면 예측이 맞았다고 판단한 지표이며 2의 경우는 두 번째로 높은 확률이라도 맞았다면 모델의 예측이 맞았다고 판단하여 정확도를 의미함
- 10 epochs 학습이 되었을 때 적중률은 각각 68%, 82.6%의 가장 좋은 성능을 도출함
- 표 8은 카테고리 분류 모듈의 confusion matrix를 통해 얻은 수치로 accuracy(정확도), recall(재현율), precision(정밀도), 그리고 F1-Score를 계산하여 나타낸 것임
- 네트워크의 재현율은 학습이 진행될수록 증가하지만 정밀도는 반대도 떨어지는 추세를 보였다. 10 epochs 학습되었을 때 정밀도와 재현율 간의 trade-off 되는 것의 최선의 결과가 나오고 이는 F1-Score를 통해 확인 가능함
- 표 7와 8의 결과를 토대로 10 epochs 학습 시 카테고리의 예측 성능이 무척 뛰어남을 알 수 있음
- 또한 그림 19과 같이 이미지의 실제 해시태그와 특징 추출 모듈을 통해 예측된 카테고리, 카테고리 분류모듈을 통해 예측된 카테고리를 확인할 수 있음

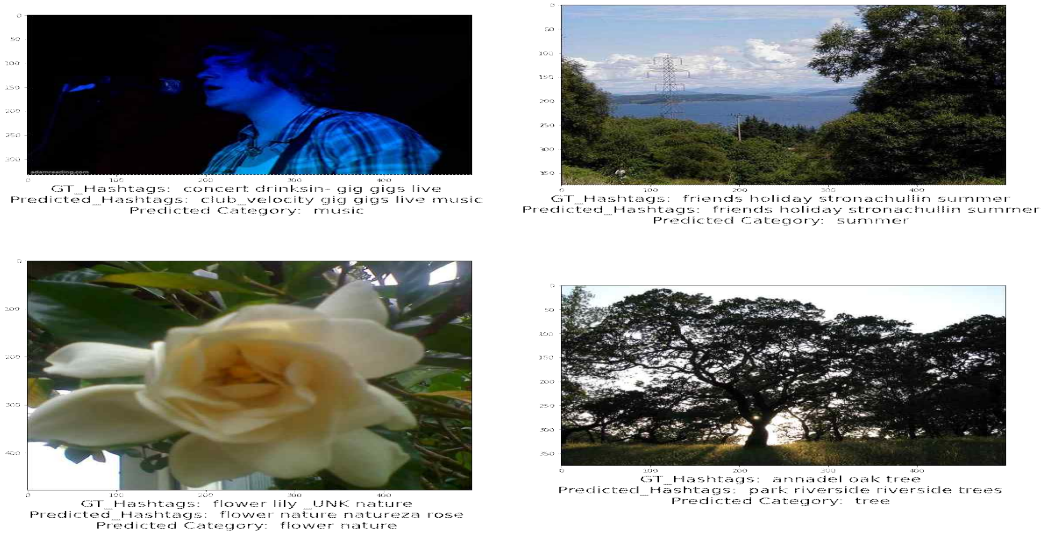


그림 31. 이미지 별 실제 해시태그와 생성된 해시태그 그리고 분류된 카테고리

(11) 알고리즘 코드 실행환경 및 사용방법

(가) 구동 하드웨어 환경

- OS: Ubuntu 18.04
- CPU: Intel Xeon(R) Gold 5220 CPU @ 2.20Ghz, 88GB RAM
- GPU: Tesla V100-SXM2-32GB

(나) 코드 실행 방법

- 아나콘다 가상환경 설치
- 해당 알고리즘이 개발된 환경과 동일한 환경을 구성하기 위해 다음의 과정을 진행함
- 아나콘다 가상환경 프로그램 설치 파일 다운로드 및 실행

```
$ wget https://repo.anaconda.com/archive/Anaconda3-2019.10-Linux-x86_64.sh
$ chmod 755 Anaconda3-2019.10-Linux-x86_64.sh

$ ./Anaconda3-2019.10-Linux-x86_64.sh
```

- 위 과정 이후에 라이센스를 확인하는 창이 나옴
- 계속 엔터를 치다보면 라이센스를 확인했냐는 질문이 나오는데 그때 y를 입력함
- 그 후에는 아나콘다르르 설치할 위치를 지정해달라는 문구가 나오는데 기본 설정으로 설치하고 싶다면 엔터를 다른 경로에 설정하고 싶다면 직접 경로를 입력함.
- PATH 등록

```
$ source ~/.bashrc
```

- 아나콘다는 기본적으로 PATH를 자동으로 등록하기에 위의 명령어로 bashrc를 최신화
- 하지만 가끔 PATH가 제대로 등록되지 않는 경우가 존재함
- 그럴 경우 아래의 방법을 시도함

```
$ sudo vi ~/.bashrc
```

- 위의 코드를 실행하면 bashrc파일을 편집할 수 있음

- I 를 눌러 편집모드로 들어가고 제일 마지막줄에 아래 문장을 추가함

```
export PATH="/home/{username}/anaconda3/bin:$PATH"
```

- 그 후 esc를 눌러 명령모드로 진입후 :wq 를 타이핑하여 저장하고 종료함
- PATH 등록이 완료되면 다시 bashrc를 최신화하여 아나콘다 프롬프트가 실행되는지 확인함

```
$ cd Classifyer_Irislab
$ conda env create -f environment.yaml
$ conda activate ClassifyNet
```

- 이 과정이 끝나면 깃허브에 별도로 올려놓은 이미지와 json파일을 다운로드함
- 또한 그것을 data_yfcc폴더에 삽입 후 압축을해제함

```
$ cd data_yfcc
$ tar -xvf yfcc_json.tar.gz
$ tar -xvf yfcc_images.tar.gz
```

- 사전 훈련된 레지듀얼 네트워크 모델을 다운받고 그것으로 데이터의 특징벡터를 추출함. 추출과 동시에 데이터의 전처리 작업도 진행함

```
$ cd ../scripts
$ ./download_pretrained_resnet_101.sh
$ ./extract_yfcc_features.sh
$ cd ..
```

- 그후 아래 명령어를 통하여 모델의 학습 및 평가를 진행할 수 있음

```
Train 시 : $ ./train.sh
Test  시 : $ ./eval.sh
```

- 위의 Train 결과는 checkpoints 폴더 안에 저장됨
- 깃허브에서 별도로 올려 둔 링크를 통하여 사전훈련된 모델을 다운받아 바로 평가또한 가능함
- Default 세팅은 하나의 gpu를 사용한 학습이며 5epoch 마다 학습이 저장됨
- model.ckpt 라는 이름으로 백본 네트워크 학습결과가 저장되고 model2.ckpt라는 이름으로 최종 분류 네트워크의 학습결과가 저장됨

3. 연구개발 성과에 대한 고찰 및 기대 효과

가. 성과에 대한 고찰

- 사용자 입력 데이터를 활용한 시각화 방법론을 구축하고 시각화한 자료를 바탕으로 다양한 알고리즘을 적용하여 여러 인공지능 알고리즘 실험 및 성능 평가 수행함
- 사용자 분류 작업에서 기대 이상의 성능을 달성하며 시각화 방법론의 적합성 발견
- 설문조사를 통한 데이터 수집 자료를 기반으로 데이터 분석을 수행하고 해당 자료로 랜덤 매핑 알고리즘으로 사용자의 패턴 분석 수행
- 사용자 특성 기반 application 연구를 진행하며 직접 모델의 알고리즘을 설계하고 생성한 해쉬태그를 기반으로 주어진 이미지가 어떤 카테고리의 게시글로 작성하였는지 예측하는 모델의 성능을 일정 수준 이상 달성함
- 사용자의 입력, 타이핑 패턴을 바탕으로 다양한 연구를 수행하였고 향후 한 가지의 연구를 선택하여 심도 깊은 연구가 필요함

나. 기대 효과

(1) 인력 양성 측면의 효과

- 인공지능 기반의 다양한 시스템 구축을 통해 인공지능 자체의 새로운 매커니즘을 만들어낼 수 있는 시각화 전문가 양성
- 개인 특정화가 가능한 정보를 분석하고 이를 익명화할 수 있는 보안 관련 융합형 인공지능 인재 양성
- 다양한 시계열 데이터를 시각화하는 과정을 통해 시계열 데이터를 다룰 수 있는 데이터 분석 전문가 양성

(2) 국가 기술 확보 측면의 효과

- 기존 영어 혹은 라틴어 등을 기반으로 연구/개발되고 있는 타이핑 기반 알고리즘을 한글로 확장하여 한글에서의 적용 가능성 확인
- 해킹 기술 및 개인정보 누출을 방지하여 정보 보안 강화 효과
- 한글 키보드의 효율성 증대 방안을 모색하고, 이를 해결하기 위한 소프트 키보드 적용 가능성 증대