kidp 한국디자인진흥원

AI를 통한 치매 환자의 기억 유지 프로그램

새침해 임창한 신수현

CONTENTS

kidp

한국디자인진흥원

늘어나는 치매 환자 수

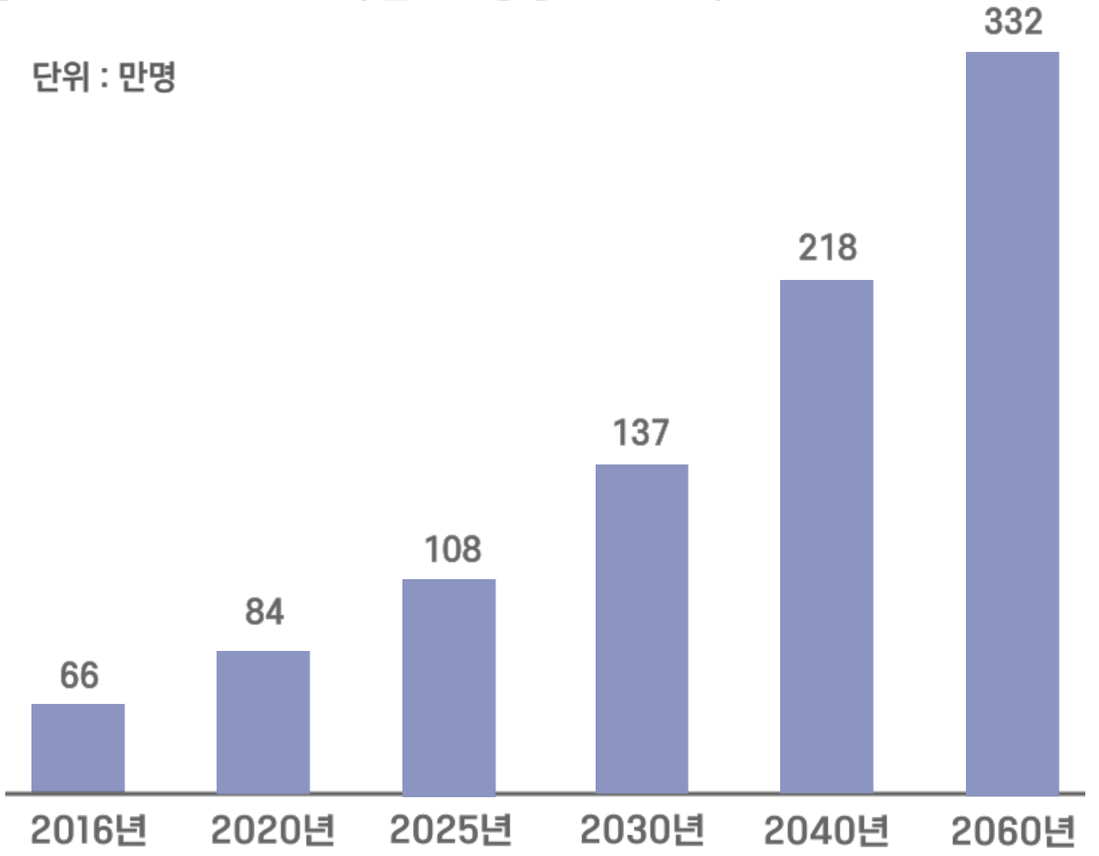
치매 환자 수는 매년 꾸준히 증가

2020년 65세 이상 노인 중
치매환자 수가 84만명에 달하는 것으로 나타났다.

65세 이상 치매 유병률은 10.3%로
노인 인구 10명 중 1명은 치매를 앓고 있었다.

평균 수명 증가 등으로 인해 노인 인구의 비중이 점차 증가함에 따라
치매환자의 수는 앞으로도 급격하게 늘어날 것으로 예측된다.

장래 치매환자 수 추이 (출처 : 중앙치매센터)



치매 미술치료의 효과

미술치료는 치매에 효과적이다

실험집단의 사전.사후검사에서
빈도분포를 양적 분석한 결과 인지기능이 향상되었고
질적 분석에서도 사전검사에 비해 형태.공간 인지력이 향상되었다.

미술 치료는 치료적인 중재를 지속할 수 있는
인지적 자극 훈련과 지속적인 손의 감각자극이
치매노인의 인지기능과 소근육 운동 기능을 향상시키는데 긍정적이었다.

치매환자 미술치료 사전·사후 점수 (출처 : 명지대학교)

점수가 낮아지는 것은 증상이 완화됨을 의미함

항목	사전		사후		F
	평균	표준편차	평균	표준편차	
반응	3.80	.447	1.00	.707	.086
과거사건	2.80	1.789	.80	.837	4.020
즐거움	2.60	.548	.40	.548	.000
유머	3.60	.548	2.00	1.000	2.415
발언	1.60	1.817	.40	.548	12.755**
의욕	3.80	.447	.60	.894	3.571
대상흥미	3.80	.447	.80	.837	1.969
의사소통	3.20	.837	.60	.894	.094
소리	2.00	2.000	.40	.548	7.724*
얼굴표정	3.20	.837	.20	.447	1.969
시선	3.40	.894	.20	.447	3.571
제스쳐	2.40	1.517	.60	.894	3.881

치매를 예방 · 치료 할 수 있는 미술 활동 활성화

미술치료 프로그램의 필요성

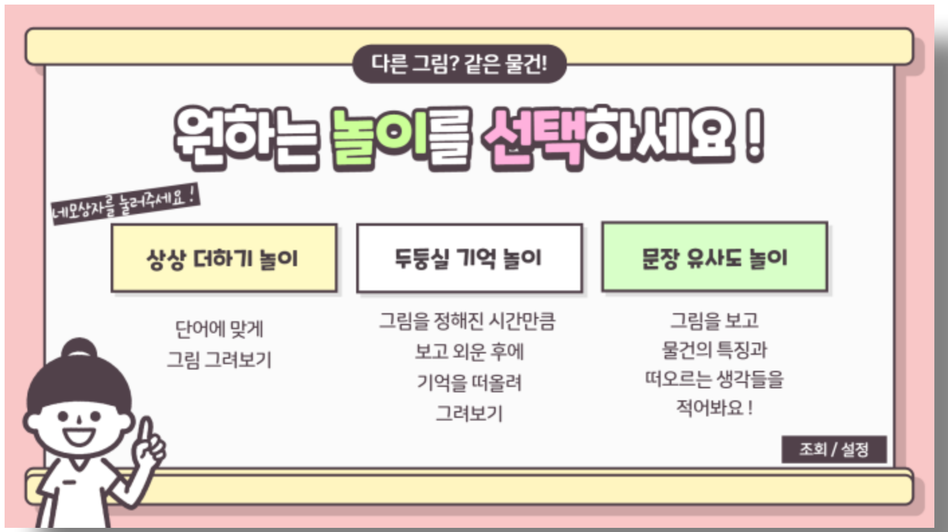
코로나 19 장기화로 사회활동이 줄고 일상생활이 변하면서
노인들의 치매 발병율이 더 높아지고 있다.

지역의 치매지원센터나 전문 병원에서 치매 진단 여부를
확인하지만 전문가들의 대면 치료가 쉽지 않다 보니
비대면 치료 프로그램이 필요하다.

특히 치매 초기와 중기 단계의 경우 일상생활에서 하던
취미 활동은 물론 치료 프로그램 활동마저 축소 돼
뇌 기능이 더 떨어질 수 밖에 없다.

비대면으로 예방 · 치료 할 수 있는 애플리케이션

게임 화면



애플리케이션의 첫 화면

게임 1 : 상상 더하기 놀이

단어를 하나 제시해주면 사용자가 상상력을 이용하여
그에 맞는 그림을 그려보는 활동

게임 2 : 두둥실 기억 놀이

일상생활에서 쉽게 볼 수 있는 물체의 일러스트를 60초간 보고
기억을 되살려 똑같이 그려보는 활동

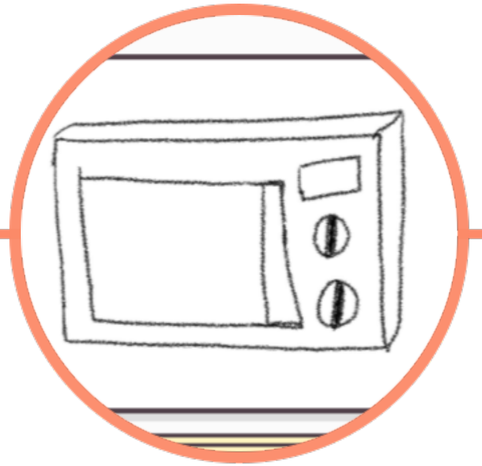
게임 3 : 문장 유사도 놀이

물체의 픽토그램과 명칭을 알려주고 주어진 조건에 맞게
질문에 문장으로 대답을 하는 활동

조회 / 설정 버튼

사용자의 정보와 약 일주일 간의 게임 내역, 정답률을 확인
언어와 글씨 크기 등 사용자에게 맞게 설정 가능

상상 더하기 놀이



1. 사용자에게 단어 제시

사용자에게 데이터셋에
존재하는 사물들 중에
하나를 랜덤으로 제시한다

2. 단어를 보고 그리기

제시된 단어를 보고
기억속의 이미지를
상상하여 그린다

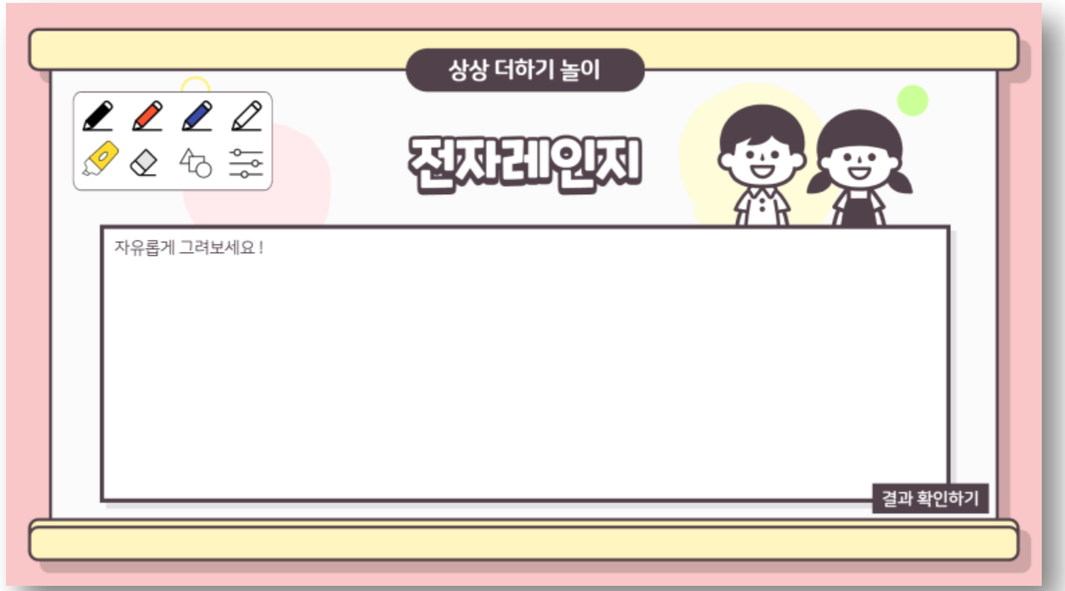
3. 그림의 라벨링 판별

학습된 모델을 통해
그려진 이미지의
라벨링을 도출한다

4. 정답 제시

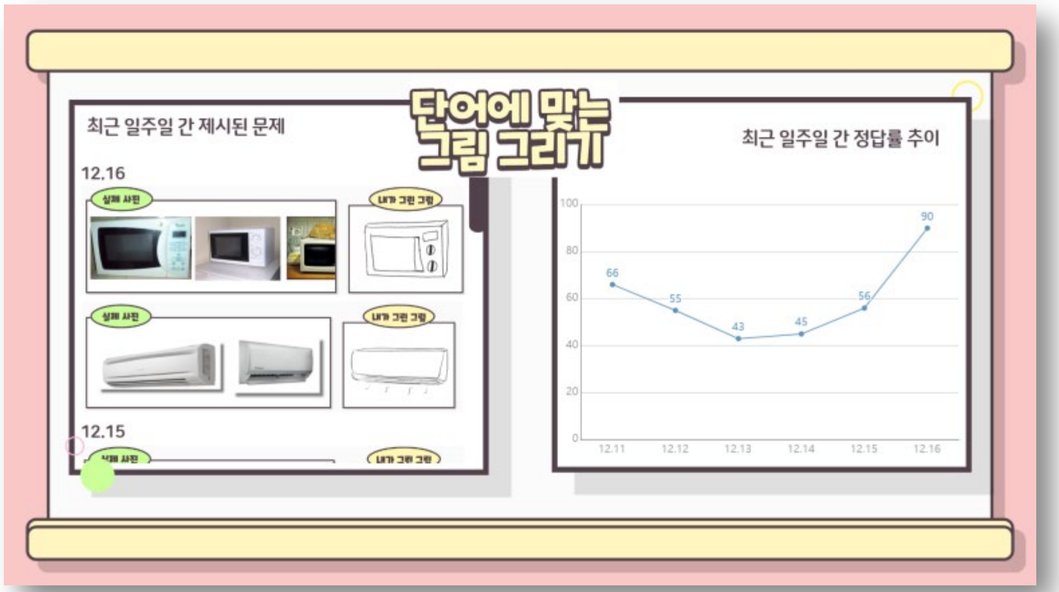
그림의 라벨링과
실제 주어진 단어가
동일한지 사용자에게 알려준다

상상 더하기 놀이 UI


상상 더하기 놀이
전자레인지
자유롭게 그려보세요 !
결과 확인하기


상상 더하기 놀이
전자레인지
자유롭게 그려보세요 !
결과 확인하기


결과 확인하기
정답입니다 !
실제 사진
내가 그린 그림
다른 놀이 선택


단어에 맞는 그림 그리기
최근 일주일 간 제시된 문제
12.16
12.15
최근 일주일 간 정답률 추이
66
55
43
45
56
90
12.11
12.12
12.13
12.14
12.15
12.16

상상 더하기 놀이 Dataset

- 사용자가 그린 그림의 대조군은 일러스트 이미지와 스케치 이미지가 적합
- 실제 이미지인 자연 이미지 데이터는 모델의 성능을 저하시킴을 확인함

< 사물의 추상 일러스트 이미지 >

< 사물의 추상 스케치 이미지 >

< 사물의 실사 이미지 >

상상 더하기 놀이 Dataset 증강

- 주어진 데이터 셋이 부족하여 데이터를 증강할 필요가 있다고 판단
- 우선적으로 30개의 카테고리에 대하여 데이터셋 증강

< Google 이미지 크롤링 >

추상 스케치 이미지와

일러스트레이션에 대하여

기존의 데이터셋과 유사한

이미지들을 수집함

< 파이썬 keras : ImageGenerator >

데이터 학습을 쉽게하도록

ImageDataGenerator 를 통해

각 파라미터들을 설정하여

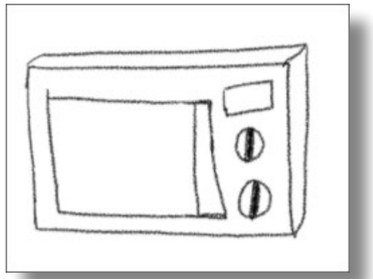
데이터셋을 증강함

상상 더하기 놀이 Model

kidp 한국디자인진흥원

이미지 클래스 분류 모델

Input



Conv2D RELU

kernel (64 X 3 X 3) bias(32)

MaxPooling 2D

Conv2D RELU

kernel (64 X 3 X 3) bias(32)

MaxPooling 2D

Conv2D RELU

kernel (3 X 3 X 128) bias(128)

MaxPooling 2D

Conv2D RELU

kernel (3 X 3 X 128) bias(128)

MaxPooling 2D

Flatten

Dropout

Dense

Dense (512, activation = 'relu')

Dense

Dense (30 , activation = 'relu')

Softmax

이미지 클래스 분류 모델

```
model = tf.keras.models.Sequential([

  tf.keras.layers.Conv2D(64, (3,3), activation='relu', input_shape=(224, 224, 3)),
  tf.keras.layers.MaxPooling2D(2, 2),
  tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
  tf.keras.layers.MaxPooling2D(2,2),
  tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
  tf.keras.layers.MaxPooling2D(2,2),
  tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
  tf.keras.layers.MaxPooling2D(2,2),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dropout(0.5),
  tf.keras.layers.Dense(512, activation='relu'),
  tf.keras.layers.Dense(30, activation='softmax')
])
```

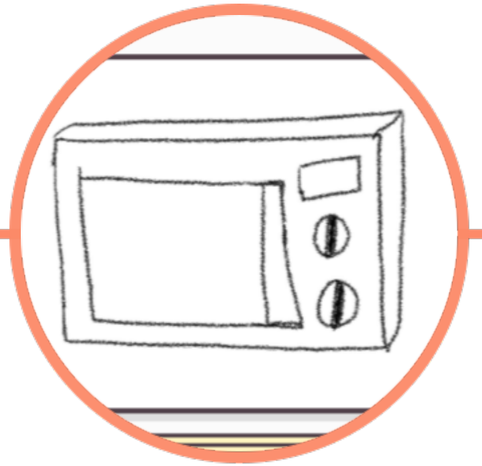
<인공지능 모델 구현 >

```
Epoch 204/250
20/20 [==============================] - 7s 349ms/step - loss: 0.6457 - accuracy: 0.6603 - val_loss: 0.6483 - val_accuracy: 0.6250
Epoch 205/250
20/20 [==============================] - 7s 352ms/step - loss: 0.6527 - accuracy: 0.6435 - val_loss: 0.7188 - val_accuracy: 0.6250
Epoch 206/250
20/20 [==============================] - 7s 347ms/step - loss: 0.6853 - accuracy: 0.5686 - val_loss: 2.6429 - val_accuracy: 0.4583
Epoch 207/250
20/20 [==============================] - 6s 280ms/step - loss: 0.7069 - accuracy: 0.5324 - val_loss: 1.5452 - val_accuracy: 0.5417
Epoch 208/250
20/20 [==============================] - 7s 323ms/step - loss: 0.7030 - accuracy: 0.5374 - val_loss: 1.8195 - val_accuracy: 0.3333
Epoch 209/250
20/20 [==============================] - 6s 309ms/step - loss: 0.6460 - accuracy: 0.6609 - val_loss: 1.5710 - val_accuracy: 0.5417
Epoch 210/250
20/20 [==============================] - 6s 295ms/step - loss: 0.6473 - accuracy: 0.6520 - val_loss: 2.5826 - val_accuracy: 0.4583
Epoch 211/250
20/20 [==============================] - 6s 281ms/step - loss: 0.6606 - accuracy: 0.6292 - val_loss: 1.1789 - val_accuracy: 0.3750
Epoch 212/250
20/20 [==============================] - 6s 301ms/step - loss: 0.6971 - accuracy: 0.5502 - val_loss: 0.6935 - val_accuracy: 0.7500
```

<인공지능 모델의 학습 과정 >

추가적인 데이터 셋과 모델 학습을 통해
전체적인 성능 향상 기대

두둥실 기억 놀이



1. 사용자에게 그림 제시

사용자에게 데이터셋에
존재하는 사물들 중에
랜덤으로 이미지 하나를 제시

2. 기억한 이미지 그리기

일정시간 동안
이미지를 기억하고
그것을 바탕으로 이미지 그리기

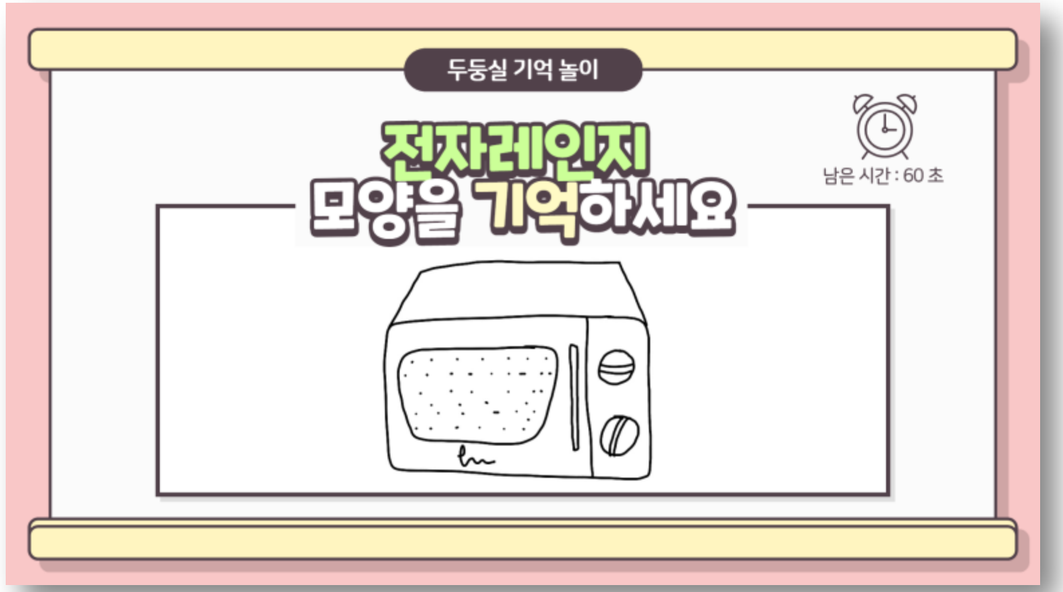
3. 그림의 유사도 측정

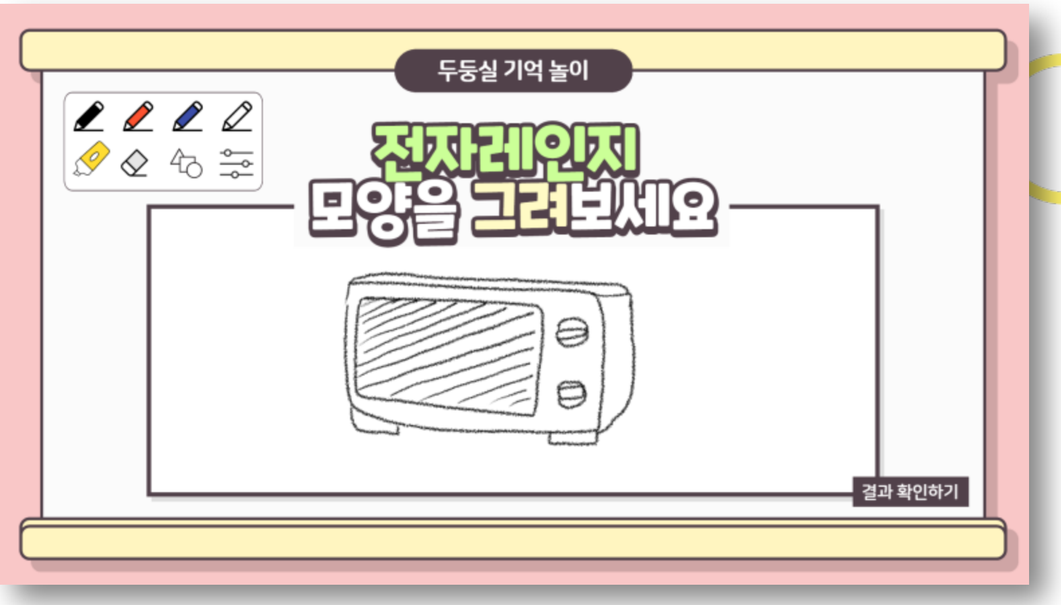
인공지능 모델과
라이브러리를 활용하여
이미지 간의 유사도 측정

4. 유사도 제시

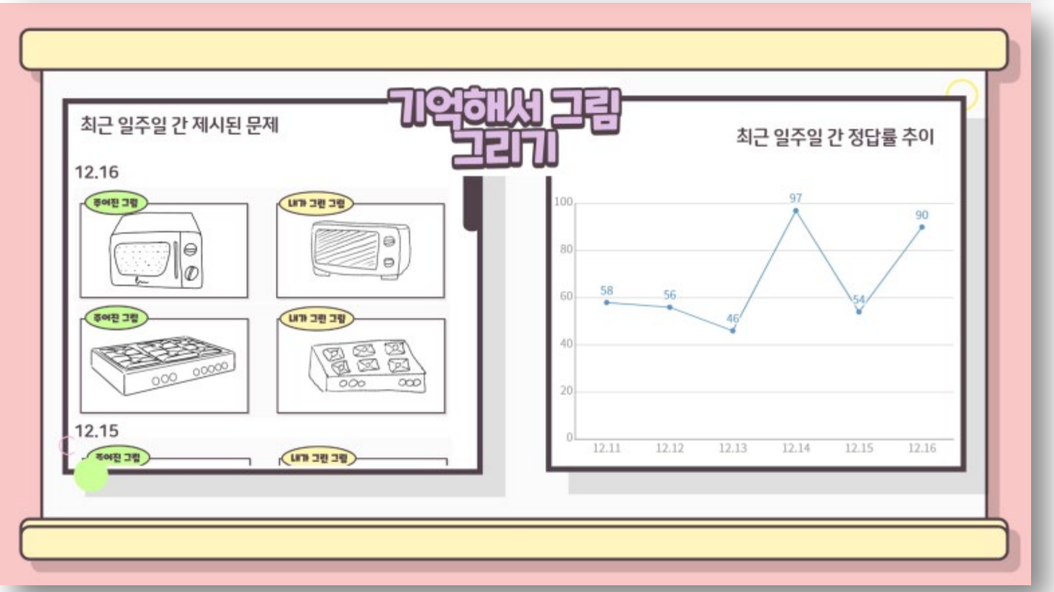
사용자에게
그림의 유사도가 몇 퍼센트인지
구체적으로 제시함

두둥실 기억 놀이 UI


두둥실 기억 놀이
전자레인지
모양을 기억하세요
남은 시간 : 60 초


두둥실 기억 놀이
전자레인지
모양을 그려보세요
결과 확인하기


결과 확인하기
오답 입니다
주어진 그림
내가 그린 그림
다른 놀이 선택

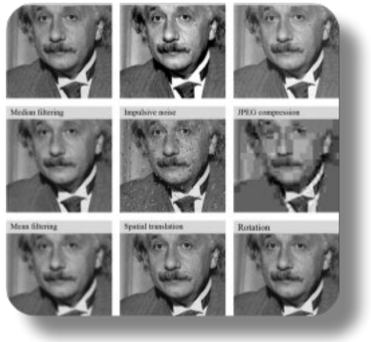

기억해서 그림
그리기
최근 일주일 간 제시된 문제
12.16
12.15
최근 일주일 간 정답률 추이
58
56
46
97
54
90
12.11
12.12
12.13
12.14
12.15
12.16

두둥실 기억 놀이

☑ 위의 구현한 인공지능 모델을 기반으로 하며 이미지 간의 유사도를 비교하는 알고리즘 추가

BFMatcher

가능한 모든 경우에 대해 다 계산해본 후 두 이미지의 유사도를 반환



SSIM(structural similarity index measure)

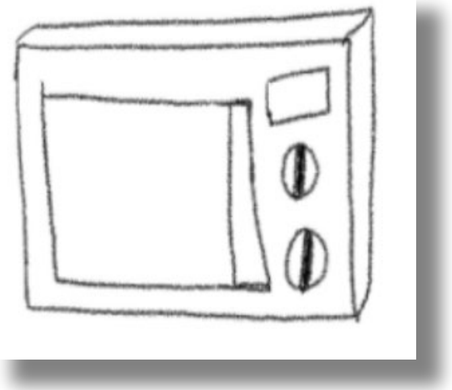
두 이미지의 휘도, 대비, 및 구조를 비교

BFMatcher

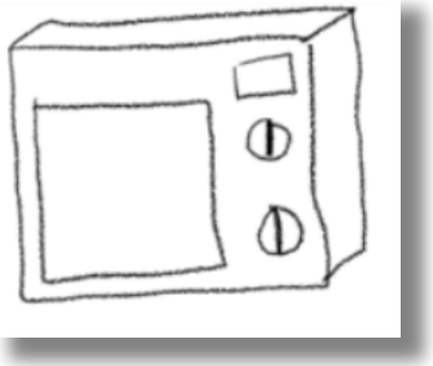
- queryDescriptors와 trainDescriptors를 하나하나 확인해 매칭되는지 판단하는 알고리즘

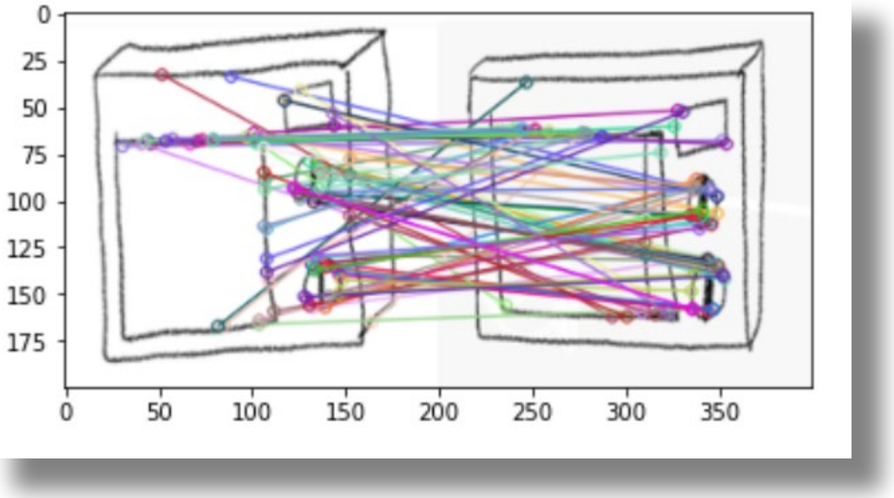
(cv2.NORM_L1, cv2.NORM_L2, cv2.NORM_L2SQR, cv2.NORM_HAMMING, cv2.NORM_HAMMING2 을 통해 거리측정)

< Image A >



< Image B >

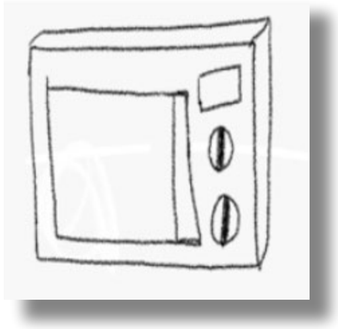




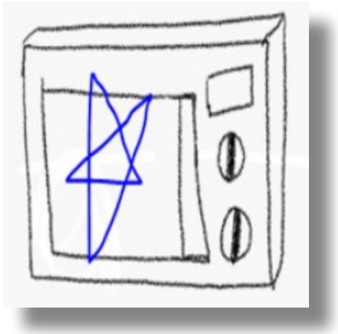
SSIM

- 원본 이미지 A와 왜곡된 이미지 B가 있다고 할 때 SSIM은 두 이미지의 휘도, 대비, 및 구조를 비교

< Image A >



< Image B >

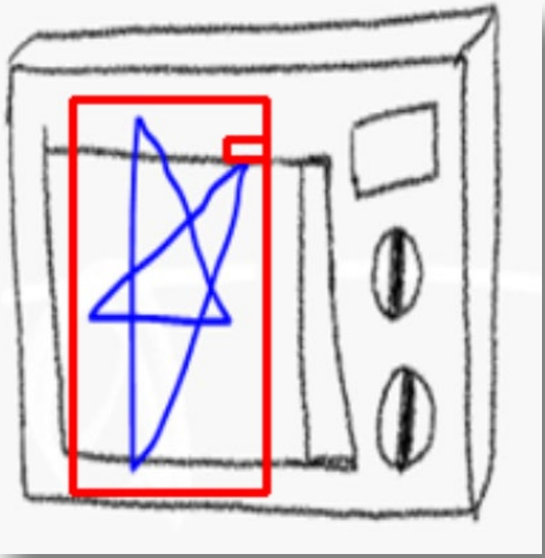


< Python Code >

```
def main():
    args = parse_args()
    imageA = cv2.imread("D:₩₩usado₩₩6.png")
    imageB = cv2.imread("D:₩₩usado₩₩2.png")


    grayA = cv2.cvtColor(imageA, cv2.COLOR_BGR2GRAY)
    grayB = cv2.cvtColor(imageB, cv2.COLOR_BGR2GRAY)
    (score, diff) = compare_ssim(grayA, grayB, full=True)
    diff = (diff * 255).astype("uint8")
    print(f"SSIM: {score}")
    thresh = cv2.threshold(
                diff, 0, 200,
                cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU
            )[1]
    cnts, _ = cv2.findContours(
                thresh,
                cv2.RETR_EXTERNAL,
                cv2.CHAIN_APPROX_SIMPLE
              )
    for c in cnts:
        area = cv2.contourArea(c)
        if area > 40:
            x, y, w, h = cv2.boundingRect(c)
            cv2.rectangle(imageA, (x, y), (x + w, y + h), (0, 0, 255), 2)
            cv2.drawContours(imageB, [c], -1, (0, 0, 255), 2)
    cv2.imshow("Original", imageA)
    cv2.imshow("Modified", imageB)
    cv2.waitKey(0)
```

< 결과 이미지 >



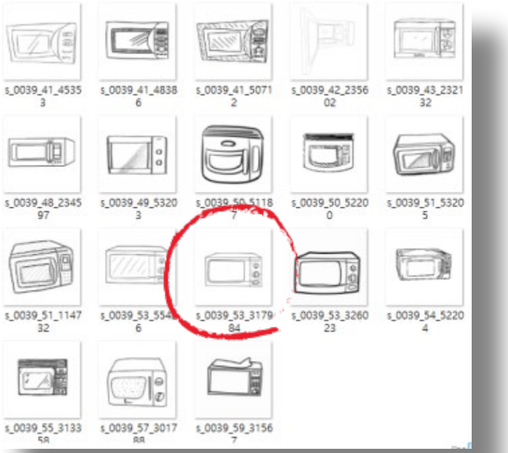
유사도 확인

- 우선적으로 인공지능 모델에서 사용자가 그린 그림을 라벨링 한 후
  해당 라벨이 맞으면 모든 이미지 별로 유사도를 측정하여 정답 확인

사용자의 그림을
인공지능 모델을 통해 라벨링

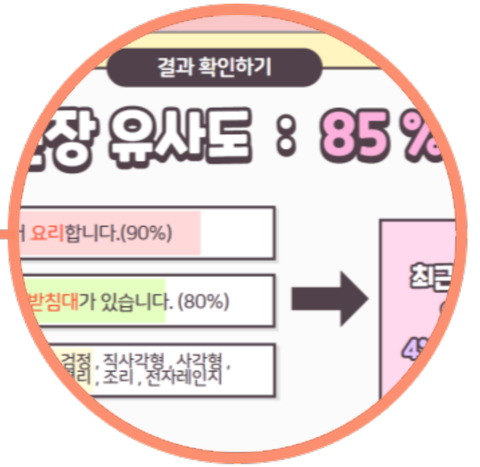
no	ctg_Id	ctg_nm_level1	ctg_nm_level2	ctg_nm_level3
1	1	가구	거실가구	장식장
2	2	가구	거실가구	TV거실장
3	3	가구	거실가구	거실테이블
4	4	가구	거실가구	사이드테이블
5	5	가구	거실가구	소파
6	6	가구	거실가구	거실의자
7	7	가구	거실가구	좌식의자
8	8	가구	거실가구	흔들의자
9	9	가구	사무/학습용가구	독서실책상
10	10	가구	사무/학습용가구	사무용 책상
11	11	가구	사무/학습용가구	일자형책상
12	12	가구	사무/학습용가구	협탁
13	13	가구	사무/학습용가구	캐비닛
14	14	가구	사무/학습용가구	책장
15	15	가구	사무/학습용가구	사무용의자
16	16	가구	야외용가구	벤치
17	17	가구	야외용가구	야외용의자(선베드)
18	18	가구	주방침실가구	주방수납장(상부, 하부장)
19	19	가구	주방침실가구	싱크대
20	20	가구	주방침실가구	침대
21	21	가구	주방침실가구	식탁
22	22	가구	주방침실가구	서랍장
23	23	가구	주방침실가구	스툴

모델의 예측값이 가장
높은 라벨 선택 후
위의 유사도 알고리즘으로 유사도 측정



라벨의 데이터 중
유사도가 가장 높은 모델로
정답 확인

문장 유사도 놀이



1. 사용자에게 그림 제시

사용자에게 그림과
그림에 대한 질문을 제시하며
특징점이 가장 두드러지는
픽토그램 이미지를 활용한다

2. 질문에 답하기

그림과 연관된
질문에 자유롭게
생각을 서술하도록 한다

3. 문장 유사도 측정

알고리즘을 통해
문장 유사도를 도출한다

4. 유사도 확인

서술한 문장과
정답 문장의
유사도를 제공한다

문장 유사도 놀이 UI

kidp 한국디자인진흥원

문장 유사도 놀이

그림 보고 생각 적어보기

전자레인지

01 어디서 어떻게 사용하는 물건일까요?

'요리'라는 단어를 넣어 설명해봅시다.

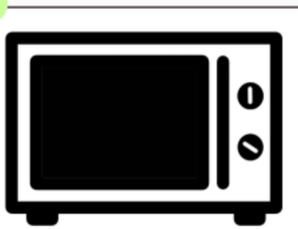
02 모양을 보고 떠오르는 생각을 적어보세요!

자유롭게 적어봐요!

결과 확인하기

문장 유사도 놀이

그림 보고 생각 적어보기



전자레인지

01 어디서 어떻게 사용하는 물건일까요?

차가운 음식을 넣어 요리합니다.

02 모양을 보고 떠오르는 생각을 적어보세요!

네모난 모양입니다. 받침대가 있습니다.

결과 확인하기

결과 확인하기

문장 유사도 : 85 %

01 차가운 음식을 넣어 요리합니다.(90%)

02 네모난 모양입니다. 받침대가 있습니다. (80%)

그 외 정답 버튼, 레버, 손잡이, 흰색, 검정, 직사각형, 사각형, 다리, 동그라미, 원, 간편, 편리, 조리, 전자레인지

최근 일주일 간 유사도가 4% 증가했어요!

다른 놀이 선택

문장 유사도 놀이

최근 일주일 간 제시된 문제

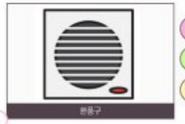
12.16

01 차가운 음식을 넣어 요리합니다.(90%)

02 네모난 모양입니다. 받침대가 있습니다. (80%)

그 외 정답 버튼, 레버, 손잡이, 흰색, 검정, 직사각형, 사각형, 다리, 동그라미, 원, 간편, 편리, 조리, 전자레인지

12.15

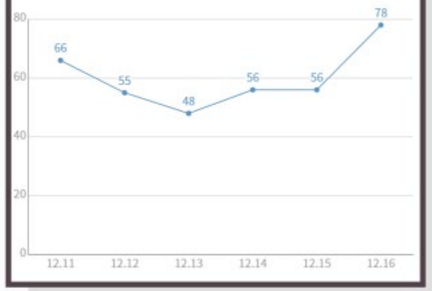


01 냉장고를 만들기 위해 사용합니다.(95%)

02 네모난 모양에 동그라미 모양이 들어가 있습니다. 동그라미 모양은 동그라미 모양입니다.(60%)

그 외 정답 네모, 시리얼, 정사각형, 동그라미, 원, 검정색, 빨간색, 음식, 화장실, 무덤, 박장, 블루진

최근 일주일 간 정답률 추이



문장 유사도 놀이

- 현재의 데이터셋은 이미지 데이터셋만 존재하나
  차후에 json 파일 형식 등 이미지에 매핑된 문장을 통해 자연어 처리 학습이 가능하다고 판단

- ☑ < Cosine > → 문장을 벡터로 변환하여 코사인 계산을 통한 유사도 측정
- ☑ < N gram > → 일부 단어만 고려하여 유사도 측정
- ☑ < Scikit-learn > → 사이킷런 (Scikit-learn) 을 통한 유사도 측정
- ☑ < TF - IDF > → TF-IDF을 통한 유사도 측정

측정 알고리즘 구현

```
from numpy import dot
from numpy.linalg import norm
import numpy as np
def cos_sim(a, b):
    return dot(a, b)/(norm(a)*norm(b))

def make_matrix(feats, list_data):
    freq_list = []
    for feat in feats:
        freq = 0
        for word in list_data:
            if feat == word:
                freq += 1
        freq_list.append(freq)
    return freq_list


okt = Okt()
text1 = '신수현 바보'
text2 = '신수현 바보'
```

Cosine

벡터 간의 코사인 각도를
이용하여 얼마나 유사한지 측정

```
def ngram(s, num):
    res=[]
    slen=len(s)-num+1
    for i in range (slen):
        ss=s[i:i+num]
        res.append(ss)
    return res

def diff_ngram(sa,sb,num):
    a=ngram(sa,num)
    b=ngram(sb,num)
    r=[]
    cnt=0
    for i in a:
        for j in b:
            if i==j:
                cnt+=1
                r.append(i)
    return cnt/len(a), r
```

N - gram

텍스트에서 이웃한 N개의 문자
서로 다른 2개의 문장을 비교하여
출현하는 단어의 종류와 빈도를 확인

Scikit-learn

형태소를 분석하여
문장의 벡터간 거리를 구함

```
import scipy as sc
contents_tokens = [t.morphs(row) for row in contents]
contents_tokens
```

```
[['신수현', '은', '찜', '갈비', '를', '먹고', '싶었지만', '곱창전골', '
 ['곱창전골', '보다', '찜', '갈비', '를', '먹고싶은', '신수현'],
 ['신수현', '은', '그냥', '매운거만', '있으면', '좋아한다'],
 ['근데', '나', '는', '분식', '이런', '거', '보다', '한식', '이', '맛있
```

```
contents_for_vectorize = []

for content in contents_tokens:
    sentence = ''      # 한 문장 문장 초기화
    for word in content:      # 단어 하나하나 띕아서 공백과 합쳐주기
        sentence = sentence + ' ' + word

    contents_for_vectorize.append(sentence)

contents_for_vectorize
```

```
[' 신수현 은 찜 갈비 를 먹고 싶었지만 곱창전골 을 먹었다',
 ' 곱창전골 보다 찜 갈비 를 먹고싶은 신수현'
```

TF-IDF

단어별로 부과하는
가중치를 통해 문서들 사이의
비슷한 정도 측정

```
def tfidf(t, d, D):
    tf = float(d.count(t)) / sum(d.count(w) for w in set(d))
    idf = sp.log(float(len(D)) / len([doc for doc in D if t in d
    return tf, idf
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=1, decode_error='ignore')
```

```
contents_tokens = [t.morphs(row) for row in contents]

contents_for_vectorize = []

for content in contents_tokens:
    sentence = ''
    for word in content:
        sentence = sentence + ' ' + word

    contents_for_vectorize.append(sentence)

X = vectorizer.fit_transform(contents_for_vectorize)
```

딥러닝을 위한 데이터 셋 구축

- 위의 알고리즘을 바탕으로 하여 사용자로부터 제공받은 문장과 정답 문장 간의 유사도 측정이 가능하지만 **데이터를 지속적으로 수집하여 딥러닝을 위한 데이터 셋 구축 가능**

	번호	날짜	점수	문장
0	1	2021-12-02	50	전자레인지는 직사각형 모양이며 타이머가 있다.
1	2	2021-12-02	45	전자레인지는 직사각형이며 타이머가 있다.
2	3	2021-12-03	55	전자레인지는 직사각형 모양이며 돌릴 수 있는 타이머가 있다.
3	4	2021-12-04	55	전자레인지는 직사각형 모양이며 돌리는 형식의 타이머가 있다.
4	5	2021-12-04	45	전자레인지는 네모 모양이며 타이머가 있다.
5	6	2021-12-04	50	전자레인지는 네모 모양이며 돌릴 수 있는 타이머가 있다.
6	7	2021-12-04	55	전자레인지는 네모 모양이며 돌리는 형식의 타이머가 있다.
7	8	2021-12-04	65	전자레인지는 직사각형 모양이며 버튼을 눌러서 문을 연다.
8	9	2021-12-03	45	전자레인지는 직사각형 모양이며 문이 있다.
9	10	2021-12-03	55	전자레인지는 직사각형 모양이며 문과 타이머가 있다.
10	11	2021-12-03	25	전자레인지는 직사각형 모양이다.
11	12	2021-12-03	25	전자레인지는 타이머가 있다.
12	13	2021-12-03	25	전자레인지는 문이 있다.
13	14	2021-12-02	25	전자레인지는 네모 모양이다.
14	15	2021-12-02	25	전자레인지는 버튼이 있다.
15	16	2021-12-02	35	전자레인지는 시간을 잴 수 있는 타이머가 있다.
16	17	2021-12-05	35	전자레인지는 버튼을 통해 문을 열 수 있다.
17	18	2021-12-06	40	전자레인지는 타이머와 버튼이 있다.
22	23	2021-12-08	45	전자레인지는 문을 열기 위해 버튼을 눌러야 한다.
23	24	2021-12-09	45	전자레인지는 버튼을 누르면 문이 열린다.
24	25	2021-12-09	55	직사각형 모양인 전자레인지는 타이머가 있다.
25	26	2021-12-09	55	직사각형 모양인 전자레인지는 버튼이 있다.
26	27	2021-12-10	65	네모 모양의 전자레인지는 타이머와 버튼이 있다.
27	28	2021-12-10	70	네모 모양의 전자레인지는 돌리는 형식의 타이머가 있다.
28	29	2021-12-11	65	직사각형 모양인 전자레인지는 타이머와 버튼이 있다.
29	30	2021-12-11	60	직사각형 모양인 전자레인지는 돌리는 형식의 타이머가 있다.
30	31	2021-12-12	60	직사각형 모양인 전자레인지는 버튼을 눌러서 문을 열 수 있다.
31	32	2021-12-12	100	직사각형 모양인 전자레인지는 돌리는 형식의 타이머가 있고 버튼을 눌러서 문을 열 수...
32	33	2021-12-13	90	직사각형 모양인 전자레인지는 타이머가 있고 버튼을 눌러서 문을 열 수 있다.
33	34	2021-12-13	100	전자레인지는 직사각형 모양이며 돌리는 형식의 타이머가 있고 버튼을 눌러서 문을 열 ...
34	35	2021-12-02	50	네모 모양의 전자레인지는 타이머가 있다.
35	36	2021-12-03	50	네모 모양의 전자레인지는 버튼이 있다.
36	37	2021-12-04	65	네모 모양의 전자레인지는 버튼을 눌러서 문을 열 수 있다.
37	38	2021-12-04	65	네모 모양의 전자레인지는 돌리는 형식의 타이머와 버튼이 있다.
38	39	2021-12-15	100	네모 모양의 전자레인지는 돌리는 형식의 타이머가 있고 버튼을 눌러서 문을 열 수 있다.
39	40	2021-12-16	45	전자레인지는 버튼을 눌러서 문을 연다.

인공지능 모델을 통한 문장 유사도 예측

- 다수의 사용자로 부터 수집된 한국어 문장을 통해 데이터셋이 충분히 수집되면 자연어 처리 인공지능 모델을 통해 사용자들에게 높은 수준의 프로그램 제공

< 문장 형태소 분석 예시 >

```
from konlpy.tag import Okt
okt = Okt()
print(okt.morphs('전자레인지는 직사각형 모양이며 돌리는 형식의 타이머가 있고 버튼을 눌러서 문을 열수있다'))
```

```
['전자레인지', '는', '직사각형', '모양', '이며', '돌리는', '형식', '의', '타이머', '가', '있고', '버튼', '을', '눌러서', '문', '을',
'열수있다']
```

< 형태소 분석, 단어의 임베딩 후 문장 학습 과정 예시 >

```
from tqdm import tqdm

X_train = []
for sentence in tqdm(train_data['reviews']):
    tokenized_sentence = okt.morphs(sentence, stem=True)
    stopwords_removed_sentence = [word for word in tokenized_sentence if not word in stopwords]
    X_train.append(stopwords_removed_sentence)
```

```
100%|██████████| 149931/149931 [20:46<00:00, 120.25it/s]
```

인공지능 모델을 통한 문장 유사도 예측

< 인공지능 모델 예시 >

```
embedding_dim = 100
hidden_units = 64

model = Sequential()
model.add(Embedding(vocab_size, embedding_dim))
model.add(LSTM(hidden_units))
model.add(Dense(1, activation='sigmoid'))
```

```
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=4)
mc = ModelCheckpoint('game3.h5', monitor='val_acc', mode='max', verbose=1, save_best_only=True)
```

```
model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
history = model.fit(X_train, y_train, epochs=15, callbacks=[es, mc], batch_size=16, validation_split=0.2)
```

< 모델의 최종적인 유사도 값 예측 >

```
sentiment_predict('전자레인지는 네모난 형태이고 손잡이가 없으며 돌리는 형식의 타이머가 있다')
print("전자레인지는 네모난 형태이고 손잡이가 없으며 돌리는 형식의 타이머가 있다 ----> 문장 유사도 : 88 %")
```

```
전자레인지는 네모난 형태이고 손잡이가 없으며 돌리는 형식의 타이머가 있다 ----> 문장 유사도 : 88 %
```

미술 활동을 통한 치매 예방 애플리케이션

기대 효과

01

쉬운 학습 방법

간단하고 쉬운
게임 학습 방법과
디자인으로
누구나 쉽게 이용 가능

02

다양한 콘텐츠

2가지 방식의
그림 그리기 콘텐츠와
문장 유사도 게임으로
다양한 콘텐츠 이용

03

불안감·우울증 완화

그림 활동을 통해
즐거운 기억을
이끌어냄으로써
불안감·우울증 완화

04

치매 예방·치료

치매 예방 및 치료에
미술 활동을 통한
긍정적 효과 기대