단어 임베딩 기법 Word2Vector 구현하기

In [ ]:

```
# 사용 모듈 import
import random
import gensim
import pandas as pd
import numpy as np
from pprint import pprint
from konlpy.tag import Okt

from tensorflow import keras
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Embedding, Dense, Softmax, Activation, Input, Dot, Reshape
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.utils import model_to_dot
from IPython.display import SVG
```

In [ ]:

```
# 데이터 읽기
d_path = './data/Chatbot_data-master/ChatbotData.csv'
df = pd.read_csv(d_path)
print(df.columns)
```

In [ ]:

```
# 데이터 확인
kor_df = df['Q']
print(kor_df)
```

In [ ]:

```
# DataFrame에서 필요한 values만 추출
kor_sentences = kor_df.values
pprint(kor_sentences[:5])
```

In [ ]:

```
# Tokenize를 위한 모듈 import 및 확인
okt = Okt()
print(okt.pos(kor_sentences[0]))
```

In [ ]:

```
# 문장별 토크나이즈
kor_word_splited = [okt.pos(sentence) for sentence in kor_sentences]
```

In [ ]:

```
# 데이터 확인
pprint(kor_word_splited[:5])
```

In [ ]:

```
# 데이터 입력, 출력 페어 생성 함수 작성
def make_pair(sentences, window_size, negative_sample):
    x_train = []
    y_train = []
    half_window_size = window_size // 2

    for sentence in sentences:
        for i in range(len(sentence)):
            if not sentence:
                continue
            if i < half_window_size:
                temp_data = sentence[0:i+half_window_size+1]
            elif i == len(sentence):
                temp_data = sentence[i-half_window_size:i+1]
            else:
                temp_data = sentence[i-half_window_size:i+half_window_size+1]

            if len(temp_data) == window_size:
                temp_idx = half_window_size
            else:
                temp_idx = len(temp_data) - (half_window_size+1)

            temp_x, temp_y = [], []
            for idx in range(len(temp_data)):
                if idx == temp_idx:
                    continue
                temp_x.append((temp_data[idx][0], temp_data[temp_idx][0]))
                temp_y.append(1)

            for j in range(negative_sample):
                negative_sentence = sentence.copy()
                for item in temp_data:
                    negative_sentence.remove(item)
                if not negative_sentence:
                    continue
                random_number = random.randrange(0, len(negative_sentence))
                negative_word = negative_sentence[random_number]
                for idx in range(len(temp_data)):
                    if idx == temp_idx:
                        continue
                    temp_x.append((temp_data[idx][0], negative_word[0]))
                    temp_y.append(0)
            x_train.append(temp_x)
            y_train.append(temp_y)
    return x_train, y_train
```

In [ ]:

```
# 윈도우 사이즈 및 네거티브 샘플 개수 설정
window_size = 3
negative_sample = 1
```

In [ ]:

```
# 입력, 출력 나누기
x_train, y_train = make_pair(kor_word_splited, window_size, negative_sample)
```

In [ ]:

```
# words_set 생성 및 확인
words_set = list(sorted(set([word for sentence in kor_word_splited for word, pos in sentence])))
print(len(words_set))
print(words_set[:5])
print(words_set[-5:])
```

In [ ]:

```
# 토크나이즈된 문장 확인
print(kor_word_splited[0])
```

In [ ]:

```
# 입력 출력 페어 확인
pprint(x_train[:5])
pprint(y_train[:5])
```

In [ ]:

```
# 단어 임베딩 사이즈 및 전체 단어 개수 설정
embedding_size = 50
vocab_size = len(words_set)
```

In [ ]:

```
# 주변 단어 인풋 레이어 생성
x_inputs = Input(shape=(1,), dtype='int32')
x_embedding = Embedding(vocab_size, embedding_size)(x_inputs)
```

In [ ]:

```
# 타겟 단어 인풋 레이어 생성
y_inputs = Input(shape=(1,), dtype='int32')
y_embedding = Embedding(vocab_size, embedding_size)(y_inputs)
```

In [ ]:

```
# 두 벡터 dot product 연산 및 출력 레이어 생성
dot_product = Dot(axes=2)([x_embedding, y_embedding])
dot_product = Reshape((1,), input_shape=(1, 1))(dot_product)
output = Activation('sigmoid')(dot_product)
```

In [ ]:

```
# 모델 생성
model = Model(inputs=[x_inputs, y_inputs], outputs=output)
model.summary()
```

In [ ]:

```
# 모델 시각화
SVG(model_to_dot(model, show_shapes=True, dpi=65).create(prog='dot', format='svg'))
```

In [ ]:

```
# 모델 컴파일
model.compile(loss='binary_crossentropy', optimizer='adam')
```

In [ ]:

```
# 모델 학습
for epoch in range(1, 101):
    print('Epoch: ', epoch, end=' ')
    loss = 0
    for x, y in zip(x_train[:50], y_train[:50]):
        x1 = np.asarray([words_set.index(word) for word, _ in x]).astype('int32')
        x2 = np.asarray([words_set.index(word) for _, word in x]).astype('int32')
        y = np.asarray(y).astype('int32')
        loss += model.train_on_batch([x1, x2], y)
    print('Loss: ', loss)
```

In [ ]:

```
# 단어 벡터 포맷 맞춰서 파일에 쓰기
f = open('w2v.txt', 'w', encoding='utf-8')
f.write('{} {}₩n'.format(vocab_size-1, embedding_size))
vectors = model.get_weights()[0]
for i, word in enumerate(words_set):
    f.write('{} {}₩n'.format(word, ' '.join(map(str, list(vectors[i, :])))))
f.close()
```

In [ ]:

```
# gensim 모듈로 벡터 읽기
w2v = gensim.models.KeyedVectors.load_word2vec_format('./w2v.txt', encoding='utf-8', binary=False)
```

In [ ]:

```
# 벡터 모델 확인
w2v.most_similar(positive=['학교'])
```

In [ ]:

```
# 벡터 모델 확인
w2v.most_similar(positive=['지망'])
```

In [ ]:

```
# 벡터 모델 확인
w2v.most_similar(positive=['물증'])
```

In [ ]:

```
# 벡터 모델 확인
w2v.most_similar(positive=['허전하다'])
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```