단어 임베딩 기법 FastText 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 jamo import h2j, j2h

from tensorflow import keras
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Embedding, 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 [ ]:

```
# 사용할 데이터만 추출
kor_sentences = list(kor_df.values)[:50]
kor_sentences[:5]
```

In [ ]:

```
# 문장 전처리
okt = Okt()
kor_word_splited = [okt.pos(sentence) for sentence in kor_sentences]
```

In [ ]:

```
# words_set 생성
def make_words_set(sentences):
    words_set = set()
    for sentence in sentences:
        for word, pos in sentence:
            if pos == "Punctuation":
                continue
            print(word)
            words_set.add(word)
    return list(sorted(words_set))
```

In [ ]:

```
# words_set 확인
words_set = make_words_set(kor_word_splited)
```

In [ ]:

```
# words_set 확인
print(len(words_set))
print(words_set[:5])
print(words_set[-5:])
```

In [ ]:

```
# 단어->자음 모음 n-gram 변환 함수 작성
def word2ngram(word, n):
    ngram = []
    ws = '<'+h2j(word)+'>'

    for i in range(len(ws)-(n-1)):
        ngram.append(ws[i:i+n])
    return ngram
```

In [ ]:

```
# tri-gram_set 및 word->tri-gram 딕셔너리 생성
tri_gram_set = set()
tri_gram = {}
for word in words_set:
    ngram = word2ngram(word, 3)
    tri_gram[word] = ngram
    tri_gram_set.update(ngram)
tri_gram_set = list(sorted(tri_gram_set))
```

In [ ]:

```
# tri-gram 딕셔너리 및 길이 확인
pprint(tri_gram)
max_len = (max([len(gram) for gram in tri_gram.values()]))
print(max_len)
```

In [ ]:

```
# tri-gram set 확인
print(len(tri_gram_set))
print(tri_gram_set[:3])
```

In [ ]:

```python
# 데이터 입력, 출력 페어 생성 함수 작성
def make_pair(sequence, window_size, negative_sample, tp='sent'):
    x, y = [], []
    half_window_size = window_size // 2

    if not sequence:
        raise ValueError('No data in sequence')

    for i in range(len(sequence)):
        if i < half_window_size:
            temp_data = sequence[0:i+half_window_size+1]
        elif i == len(sentence):
            temp_data = sequence[i-half_window_size:i+1]
        else:
            temp_data = sequence[i-half_window_size:i+half_window_size+1]

        if len(sequence) < window_size:
            temp_idx = i
        elif len(temp_data) == window_size:
            temp_idx = half_window_size
        else:
            temp_idx = len(temp_data) - (half_window_size+1)

        for idx in range(len(temp_data)):
            if idx == temp_idx:
                continue
            x.append((temp_data[idx], temp_data[temp_idx]))
            y.append(1)

        for _ in range(negative_sample):
            negative_sentence = sequence.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
                x.append((temp_data[idx], negative_word))
                y.append(0)
    return x, y
```

In [ ]:

```
# 데이터 페어 생성 및 확인
NX, NY = [], []
for idx, sentence in enumerate(kor_sentences):
    words = [word for word, _ in okt.pos(sentence) if word in words_set]
    x, y = make_pair(words, window_size=3, negative_sample=1)

    for word in words:
        ngram = tri_gram[word]
        n_x, n_y = make_pair(ngram, window_size=3, negative_sample=1)
        NX += n_x
        NY += n_y

    if idx == 0:
        print(NX)
        print(NY)
```

In [ ]:

```
# 파라미터 설정
vocab_size = len(words_set)
tri_gram_size = len(tri_gram_set)
embedding_size = 50
```

In [ ]:

```
# 모델 생성
sub_x_inputs = Input(shape=(1,), dtype='int32')
sub_x_embedding = Embedding(tri_gram_size, embedding_size)(sub_x_inputs)

sub_y_inputs = Input(shape=(1,), dtype='int32')
sub_y_embedding = Embedding(tri_gram_size, embedding_size)(sub_y_inputs)

sub_dot_product = Dot(axes=2)([sub_x_embedding, sub_y_embedding])
sub_dot_product = Reshape((1,), input_shape=(1, 1))(sub_dot_product)
sub_output = Activation('sigmoid')(sub_dot_product)

sub_model = Model(inputs=[sub_x_inputs, sub_y_inputs], outputs=sub_output)
sub_model.summary()
```

In [ ]:

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

In [ ]:

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

In [ ]:

```
# 모델 학습
for epoch in range(1, 11):
    print('Epoch: ', epoch, end=' ')
    loss = 0
    for x, y in zip(NX, NY):
        x1 = np.asarray(tri_gram_set.index(x[0])).astype('int32').reshape(1, -1)
        x2 = np.asarray(tri_gram_set.index(x[1])).astype('int32').reshape(1, -1)
        y = np.asarray(y).astype('int32').reshape(1, -1)
        loss += sub_model.train_on_batch([x1, x2], y)
    print('Loss: ', loss)
```

In [ ]:

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

In [ ]:

```
# gensim으로 벡터 읽기
ft = gensim.models.KeyedVectors.load_word2vec_format('./sub_fasttext.txt', encoding='utf-8', binary=
```

In [ ]:

```
# 단어는 sum(n-gram[단어]) 이므로, 단어 벡터 생성 함수 작성
def get_word_vector(word):
    res = []
    for gram in word2ngram(word, 3):
        try:
            res.append(ft[gram])
        except KeyError:
            continue
    return np.sum(res, axis=0)
```

In [ ]:

```
# 단어별 벡터 확인
word = '학교'
word_vector_1 = get_word_vector(word)
print(word_vector_1)
```

In [ ]:

```
# 단어별 벡터 확인
word = '핵교'
word_vector_2 = get_word_vector(word)
print(word_vector_2)
```

In [ ]:

```
# 코사인 유사도 함수 작성(-1~1: -1: 전혀 다름, 1: 완전 동일)
def cos_sim(A, B):
       return np.dot(A, B)/(np.linalg.norm(A)*np.linalg.norm(B))
```

In [ ]:

```
# '학교' 과 '핵교'의 유사도
cosim = cos_sim(word_vector_1, word_vector_2)
print(cosim)
```

In [ ]:

```
# '한글'의 벡터
word = '한글'
word_vector_3 = get_word_vector(word)
print(word_vector_3)
```

In [ ]:

```
# '학교'과 '한글'의 유사도
cosim = cos_sim(word_vector_1, word_vector_3)
print(cosim)
```

In [ ]:

```
# '딥러닝'의 벡터
word = '딥러닝'
word_vector_4 = get_word_vector(word)
print(word_vector_4)
```

In [ ]:

```
# '학교'과 '딥러닝'의 유사도
cosim = cos_sim(word_vector_1, word_vector_4)
print(cosim)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: