**단어 임베딩 실습**

**1. 영어단어 임베딩**

- 쿠텐베르크 이용

**import** **requests**

**import** **re**

res = requests.get('https://www.gutenberg.org/files/2591/2591-0.txt')

**앞과 뒤에서 불필요한 부분을 완전 제외한다**.

grimm = res.text[2801:530661]

**정규표현식을 이용하여 불필요한 단어는 제외한다.**

grimm = re.sub(r'[^a-zA-Z\. ]', ' ', grimm)

**.split('. ')을 이용하여 마침표 단위로 자르자.**

sentences = grimm.split('. ') *# 문장 단위로 자름*

**이번엔 단어(빈칸) 단위로 자르도록 하자**.

data = [s.split() **for** s **in** sentences]

**단어 단위로 자른 결과를 확인해보자.**

data[0] *# 첫 번째 문장을 단어 단위로 자른 결과를 확인하자*

**- word2vec**

**이제 word2vec 를 필요한 gensim 패키지를 설치해보자**.

!conda install -y gensim

**gensim 패키지에서 Word2Vec을 불러오자.**

**from** **gensim.models.word2vec** **import** Word2Vec

**학습 하기**

model = Word2Vec(data, *# 리스트 형태의 데이터*

sg=1, *# 0: CBOW, 1: Skip-gram*

size=100, *# 벡터 크기*

window=3, *# 고려할 앞뒤 폭(앞뒤 3단어)*

min\_count=3, *# 사용할 단어의 최소 빈도(3회 이하 단어 무시)*

workers=4) *# 동시에 처리할 작업 수(코어 수와 비슷하게 설정)*

**저장 및 불러오기**

.save()를 통해 word2vec 모델을 저장할 수 있다.

model.save('word2vec.model')

저장한 모델을 불러 올 때는 .load()를 사용한다.

model = Word2Vec.load('word2vec.model')

**wv 메소드로 벡터로 바꾸기**

model.wv['princess']

wv.similarity()에 두 단어를 넘겨주면 코사인 유사도를 구할 수 있다.

model.wv.similarity('princess', 'queen') #시멀러리

-> prince와 king 거리 확인

wv.most\_similar()에 단어를 넘겨주면 가장 **유사한 단어를 추출**할 수 있다.

model.wv.most\_similar('princess') #시멀러

.wv\_most\_similar() 에 **positive와 negative라는 옵션**을 넘겨줄 수 있다.

"woman:princess = man:?"의 유비를 풀어보자.

model.wv.most\_similar(positive=['man', 'princess'], negative=['woman'])

**케라스에서 임베딩 레이어에 적용**

**from** **keras.models** **import** Sequential

**from** **keras.layers** **import** Embedding

NUM\_WORDS, EMB\_DIM = model.wv.vectors.shape

gensim으로 학습된 단어 임베딩 model.wv.vectors를 케라스의 임베딩 레이어의 가중치로 사용한다.

emb = Embedding(input\_dim=NUM\_WORDS, output\_dim=EMB\_DIM,

trainable=False, weights=[model.wv.vectors])

net = Sequential()

net.add(emb)

**예측으로 gensim과 같은지 확인**

i = model.wv.index2word.index('princess')

net.predict([i])

**저장 및 불러오기**

model.wv.save\_word2vec\_format('eng\_w2v') # 모델 저장

loaded\_model = KeyedVectors.load\_word2vec\_format("eng\_w2v") # 모델 로드

**2. 한국어 임베딩**

**번에는 네이버 영화 리뷰 데이터로 한국어 Word2Vec을 만들어봅시다.**

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**import** urllib.request

**from** gensim.models.word2vec **import** Word2Vec

**from** konlpy.tag **import** Okt

**우선 네이버 영화 리뷰 데이터를 다운로드합니다.**

urllib.request.urlretrieve("https://raw.githubusercontent.com/e9t/nsmc/master/ratings.txt", filename="ratings.txt")

네이버 영화 리뷰 데이터를 데이터프레임으로 로드하고 상위 5개의 행을 출력해봅시다.

train\_data = pd.**read**\_table('ratings.txt')

train\_data[:5] # 상위 5개 출력

**print**(len(train\_data)) # 리뷰 개수 출력

200000

총 20만개의 데이터가 존재하는데, 결측값이 있는지 확인해보겠습니다.

**isnull활용 null체크**

# NULL 값 존재 유무

**print**(train\_data.isnull().**values**.any())

**True**

Dropna 활용 결측값 제거

train\_data = train\_data.dropna(how = 'any') # Null 값이 존재하는 행 제거

**print**(train\_data.isnull().**values**.any()) # Null 값이 존재하는지 확인

**False**

결측값이 삭제된 후의 리뷰 개수를 확인해보겠습니다.

**print**(len(train\_data)) # 리뷰 개수 출력

199992

이제 총 199,992개의 리뷰가 존재합니다.

**정규 표현식**을 통해서 문자열 중 한글이 아닌 경우에 대해서는 제거하는 **전처리**를 진행합니다.

# 정규 표현식을 통한 한글 외 문자 제거

train\_data['document'] = train\_data['document'].str.replace("[^ㄱ-ㅎㅏ-ㅣ가-힣 ]","")

train\_data[:5] # 상위 5개 출력

학습 시에 사용하고 싶지 않은 단어들인 **불용어를 제거하겠습니다**.

# 불용어 정의

stopwords = ['의','가','이','은','들','는','좀','잘','걍','과','도','를','으로','자','에','와','한','하다']

**진행 상황 확인 위해 사용**

from tqdm import tqdm

**형태소 분석**

# 형태소 분석기 OKT를 사용한 토큰화 작업 (다소 시간 소요)

okt = Okt()

tokenized\_data = []

**for** sentence **in** tqdm(train\_data['document']):

tokenized\_sentence = okt.morphs(sentence, stem=**True**) # 토큰화

stopwords\_removed\_sentence = [word **for** word **in** tokenized\_sentence **if** **not** word **in** stopwords] # 불용어 제거

tokenized\_data.append(stopwords\_removed\_sentence)

**토큰화 분포 확인**

# 리뷰 길이 분포 확인

print('리뷰의 최대 길이 :',max(len(l) **for** l **in** tokenized\_data))

print('리뷰의 평균 길이 :',sum(map(len, tokenized\_data))/len(tokenized\_data))

plt.hist([len(s) **for** s **in** tokenized\_data], bins=50)

plt.xlabel('length of samples')

plt.ylabel('number of samples')

plt.show()

**이제 Word2Vec으로 토큰화 된 네이버 영화 리뷰 데이터를 학습합니다.**

**from** gensim.models **import** Word2Vec

model = Word2Vec(sentences = tokenized\_data, size = 100, window = 5, min\_count = 5, workers = 4, sg = 0)

학습이 다 되었다면 Word2Vec 임베딩 행렬의 크기를 확인합니다.

# 완성된 임베딩 매트릭스의 크기 확인

model.wv.**vector**s.shape

(16477, 100)

총 16,477개의 단어가 존재하며 각 단어는 100차원으로 구성되어져 있습니다.

**'최민식'과 유사한 단어들을 뽑아봅시다.**

**print**(model.wv.most\_similar("최민식"))

**'히어로'와 유사한 단어들을 뽑아봅시다.**

**print**(model.wv.most\_similar("히어로"))

[('슬래셔', 0.8747539520263672), ('느와르', 0.8666149377822876), ('무협', 0.8423701524734497), ('호러', 0.8372749090194702), ('물의', 0.8365858793258667), ('무비', 0.8260530233383179), ('물', 0.8197994232177734), ('홍콩', 0.8120777606964111), ('블록버스터', 0.8021541833877563), ('블랙', 0.7880141139030457)]

**.wv\_most\_similar() 에 positive와 negative라는 옵션을 넘겨줄 수 있다.**

"여자:공주 = 남자:?"의 유비를 풀어보자.

model.wv.most\_similar(positive=['남자', '공주'], negative=['여자'])

"여자:공주 = 남자:?"의 유비를 풀어보자.

**3. 사전 학습 word2vec가져오기**

**- 구글 사전 훈련 word2vec 호출 해서 쓰기**

모델 다운로드 경로 : https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit

압축 파일의 용량은 약 1.5GB이지만, 파일의 압축을 풀면 약 3.3GB의 파일이 나옵니다.

**import** gensim

# 구글의 사전 훈련된 Word2Vec 모델을 로드합니다.

model = gensim.models.KeyedVectors.load\_word2vec\_format('GoogleNews-vectors-negative300.bin 파일 경로', binary=**True**)

모델의 크기(shape)를 확인해봅시다.

**print**(model.vectors.shape)

(3000000, 300)

사전 훈련된 임베딩을 사용하여 두 단어의 유사도를 계산해봅시다.

**print** (model.similarity('this', 'is'))

**print** (model.similarity('post', 'book'))

0.407970363878

0.0572043891977

단어 'book'의 벡터를 출력해봅시다.

**print**(**model**['book'])

wv.similarity()에 두 단어를 넘겨주면 코사인 유사도를 구할 수 있다.

model.wv.similarity('princess', 'queen') #시멀러리

-> prince와 king 거리 확인

wv.most\_similar()에 단어를 넘겨주면 가장 **유사한 단어를 추출**할 수 있다.

model.wv.most\_similar('princess') #시멀러

.wv\_most\_similar() 에 **positive와 negative라는 옵션**을 넘겨줄 수 있다.

"woman:princess = man:?"의 유비를 풀어보자.

model.wv.most\_similar(positive=['man', 'princess'], negative=['woman'])

**- 한글 사전 학습 word2Vec**

임베딩은 <https://github.com/Kyubyong/wordvectors> 에서 찾을 수 있다. 한국어 Word2Vec은 Korean (w)라는 링크에서 다운 받을 수 있다. ko.zip을 다운받아 압축을 풀면 50MB 가량의 ko.bin 파일이 생긴다. 이 파일은 .load로 불러들일 수 있다.

원본경로 <https://drive.google.com/file/d/0B0ZXk88koS2KbDhXdWg1Q2RydlU/view>

**import** **numpy**

kovec = Word2Vec.load('ko.bin')

"한국:서울 = 일본:?"의 유비를 풀어보자.

kovec.wv.most\_similar(positive=['일본', '서울'], negative=['한국'])