Assignment No.5

November 10, 2022

[3]:

```
import numpy as np
import tensorflow as tf

import keras.backend as K
from tensorflow.keras.models import Sequential
from keras.layers import Dense, Embedding, Lambda
from keras.utils import np_utils
from keras.preprocessing import sequence
from keras.preprocessing.text import Tokenizer
import gensim
```

[5]:

```
data=open('covid.txt','r')
corona_data = [text for text in data if text.count(' ') >= 2]
vectorize = Tokenizer()
vectorize.fit_on_texts(corona_data)
corona_data = vectorize.texts_to_sequences(corona_data)
total_vocab = sum(len(s) for s in corona_data)
word_count = len(vectorize.word_index) + 1
window_size = 2
```

[7]:

```
def cbow_model(data, window_size, total_vocab):
    total_length = window_size*2
    for text in data:
        text_len = len(text)
        for idx, word in enumerate(text):
            context_word = []
            target   = []
            begin = idx - window_size
            end = idx + window_size + 1
            context_word.append([text[i] for i in range(begin, end) if 0 <= i < text_len and i != idx])
            target.append(word)
            contextual = sequence.pad_sequences(context_word, total_length=total_length)
            final_target = np_utils.to_categorical(target, total_vocab)
            yield(contextual, final_target)
```

[8]:

```
            model = Sequential()
            model.add(Embedding(input_dim=total_vocab, output_dim=100,␣
 ↪input_length=window_size*2))
            model.add(Lambda(lambda x: K.mean(x, axis=1), output_shape=(100,)))
            model.add(Dense(total_vocab, activation='softmax'))
            model.compile(loss='categorical_crossentropy', optimizer='adam')
            for i in range(20):
              cost = 0
              for x, y in cbow_model(data, window_size, total_vocab):
                  cost += model.train_on_batch(contextual, final_target)
              print(i, cost)
```

```
0 0
1 0
2 0
3 0
4 0
5 0
6 0
7 0
8 0
9 0
10 0
11 0
12 0
13 0
14 0
15 0
16 0
17 0
18 0
19 0
```

[9]:

```
dimensions=100
```

[10]:

```
vect_file = open('vectors.txt' ,'w')
vect_file.write('{} {}\n'.format(101,dimensions))
```

[10]:

```
8
```

[11]:

```
weights = model.get_weights()[0]
for text, i in vectorize.word_index.items():
    final_vec = ' '.join(map(str, list(weights[i, :])))
    vect_file.write('{} {}\n'.format(text, final_vec))
vect_file.close()

cbow_output = gensim.models.KeyedVectors.load_word2vec_format('vectors.txt',␣
 ↪binary=False)
```

```
cbow_output.most_similar(positive=['virus'])
```

```
[11]: [('making', 0.23960739374160767),
       ('influenza', 0.1882912963628769),
       ('point', 0.18283092975616455),
       ('time', 0.1582365185022354),
       ('individual', 0.15779957175254822),
       ('however', 0.1512787938117981),
       ('difference', 0.14933599531650543),
       ('incubation', 0.1451452225446701),
       ('both', 0.1396801471710205),
       ('symptoms', 0.12513065338134766)]
```

```
[12]: cbow_output = gensim.models.KeyedVectors.load_word2vec_format('vectors.txt',␣
       ↪binary=False)
      cbow_output.most_similar(positive=['virus'])
```

```
[12]: [('making', 0.23960739374160767),
       ('influenza', 0.1882912963628769),
       ('point', 0.18283092975616455),
       ('time', 0.1582365185022354),
       ('individual', 0.15779957175254822),
       ('however', 0.1512787938117981),
       ('difference', 0.14933599531650543),
       ('incubation', 0.1451452225446701),
       ('both', 0.1396801471710205),
       ('symptoms', 0.12513065338134766)]
```