text_to_embeddings

June 9, 2025

[1]:

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense
import numpy as np
from gensim.models import Word2Vec
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from warnings import filterwarnings
filterwarnings('ignore')
```

[2]:

```
sample_text  = '''Deep learning is a branch of machine learning that uses␣
 ↪artificial neural networks to analyze data and make predictions.
                It enables computers to learn from data, much like humans, by␣
 ↪identifying patterns and features within the data.
                Deep learning models are composed of multiple layers of␣
 ↪interconnected nodes (neurons) that process information.
                Introduction to Deep Learning'''
```

1 Using NLTK and word2vec

[3]:

```
def filter_corpus_by_words(sample_text,print_ = False ):
    processed_corpus = []
    text  = nltk.tokenize.sent_tokenize(sample_text) # converting given text␣
 ↪corpus to sentences
    stop_words = set(stopwords.words('english')) # set of stopwords in english␣
 ↪language
    for doc in text:
        tokens = word_tokenize(doc.lower())
        filtered_tokens = [word for word in tokens if word.isalnum() and word␣
 ↪not in stop_words]
        processed_corpus.append(filtered_tokens)
    if print_:
        print("Processed Corpus:")
        for doc in processed_corpus:
            print(doc)
    return processed_corpus

processed_corpus = filter_corpus_by_words(sample_text)
```

[4]:

```
vectors = Word2Vec(sentences=processed_corpus, vector_size=20, min_count=1,␣
 ↪sg=0,compute_loss=True,epochs=82)
```

[5]:

```
word = 'learning'
if word in vectors.wv:
    word_vector = vectors.wv[word]
    print(f"\nWord2Vec embedding for '{word}':\n{word_vector}")
    print(f"Shape: {word_vector.shape}")

    # 4. Find similar words
    similar_words = vectors.wv.most_similar(word, topn=5)
    print(f"\nWords similar to '{word}':\n{similar_words}")
else:
    print(f"\nWord '{word}' not in Word2Vec vocabulary.")
```

```
Word2Vec embedding for 'learning':
[-0.0041298   0.00052904  0.02738201  0.04280639 -0.04486082 -0.03578677
  0.03607799  0.04926915 -0.02861241 -0.01850656  0.04098072 -0.0094584
 -0.02237132  0.03266828 -0.02282948 -0.00871675  0.01877594  0.00700811
 -0.04272503 -0.04905445]
Shape: (20,)

Words similar to 'learning':
[('process', 0.5181392431259155), ('humans', 0.49196934700012207), ('machine',
0.4343571960926056), ('interconnected', 0.39095520973205566), ('enables',
0.37636977434158325)]
```

2 Using tensorflow

[6]:

```
import tensorflow as tf
from tensorflow.keras.layers import Embedding
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
import numpy as np
```

[7]:

```
texts  = nltk.tokenize.sent_tokenize(sample_text)
print(texts)
# 2. Tokenize the text
tokenizer = Tokenizer(num_words=100,lower=True) # Consider top 100 words
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts)
sequences
```

```
['Deep learning is a branch of machine learning that uses artificial neural
networks to analyze data and make predictions.', 'It enables computers to learn
from data, much like humans, by identifying patterns and features within the
data.', 'Deep learning models are composed of multiple layers of interconnected
nodes (neurons) that process information.', 'Introduction to Deep Learning']
```

[7]:
```
[[2, 1, 8, 9, 10, 3, 11, 1, 6, 12, 13, 14, 15, 4, 16, 5, 7, 17, 18],
 [19, 20, 21, 4, 22, 23, 5, 24, 25, 26, 27, 28, 29, 7, 30, 31, 32, 5],
 [2, 1, 33, 34, 35, 3, 36, 37, 3, 38, 39, 40, 6, 41, 42],
 [43, 4, 2, 1]]
```

[8]:
```python
word_index = tokenizer.word_index
print(f"\nWord Index (Tensorflow Tokenizer):\n{word_index}")
# print(f"Sequences:\n{sequences}")
```

```
Word Index (Tensorflow Tokenizer):
{'learning': 1, 'deep': 2, 'of': 3, 'to': 4, 'data': 5, 'that': 6, 'and': 7,
'is': 8, 'a': 9, 'branch': 10, 'machine': 11, 'uses': 12, 'artificial': 13,
'neural': 14, 'networks': 15, 'analyze': 16, 'make': 17, 'predictions': 18,
'it': 19, 'enables': 20, 'computers': 21, 'learn': 22, 'from': 23, 'much': 24,
'like': 25, 'humans': 26, 'by': 27, 'identifying': 28, 'patterns': 29,
'features': 30, 'within': 31, 'the': 32, 'models': 33, 'are': 34, 'composed':
35, 'multiple': 36, 'layers': 37, 'interconnected': 38, 'nodes': 39, 'neurons':
40, 'process': 41, 'information': 42, 'introduction': 43}
```

[9]:
```python
# 3. Pad sequences to ensure uniform length
max_sequence_len = max(len(s) for s in sequences)
padded_sequences = pad_sequences(sequences, maxlen=max_sequence_len,␣
  ↪padding='post')
print(f"\nPadded Sequences with max len:{max_sequence_len}\n{padded_sequences}")



# padded_sequences is final output
```

```
Padded Sequences with max len:19
[[ 2  1  8  9 10  3 11  1  6 12 13 14 15  4 16  5  7 17 18]
 [19 20 21  4 22 23  5 24 25 26 27 28 29  7 30 31 32  5  0]
 [ 2  1 33 34 35  3 36 37  3 38 39 40  6 41 42  0  0  0  0]
 [43  4  2  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]]
```

[10]:
```python
vocab_size = len(word_index) + 1  # Add 1 for the padding token (0)
print(vocab_size)
embedding_dim = 64 # Desired dimensionality of the embedding
```

```
44
```

[11]:

```
embedding_layer = Embedding(input_dim=vocab_size,
                            output_dim=embedding_dim,
                            input_length=max_sequence_len)


embedded_sequences = embedding_layer(padded_sequences)

print(f"\nOutput of Keras Embedding Layer (shape): {embedded_sequences.shape}")
```

```
Output of Keras Embedding Layer (shape): (4, 19, 64)
```

[12]:

```
# (batch_size, sequence_length, embedding_dim)
# batch_size is the number of texts (4 in this case)
# sequence_length is max_sequence_len (19)
# embedding_dim is 16
print(f"Example embedding for the first word of the first sentence:
 ↪\n{embedded_sequences[0, 0, :]}")
```

```
Example embedding for the first word of the first sentence:
[-0.03097786 -0.01355761 -0.03380464  0.04986094  0.03843668 -0.03074837
  0.0048359   0.04037979  0.03339995 -0.02191104 -0.0089682  -0.00721304
  0.02046641  0.01346039 -0.01451672  0.01834874 -0.02754556 -0.01184278
 -0.03647103 -0.00761991 -0.00952339 -0.03455668 -0.02673699 -0.03792853
  0.04597051 -0.01492751 -0.03240017  0.03820202  0.04928113 -0.04756137
  0.04092007  0.01362092  0.01053228 -0.04305983  0.04813161 -0.01688687
  0.04349688  0.03210661 -0.01295843 -0.03543295  0.0239558   0.03186109
 -0.04616003  0.00829976  0.00840474  0.00268687  0.00433765  0.01358309
  0.03977385 -0.01771982  0.00192568  0.03902122  0.0350115   0.04065806
  0.02825171  0.0483082   0.02349967  0.01741841  0.01136048 -0.00195349
 -0.03629371  0.0294208   0.02775467  0.0027809 ]
```

[ ]: