NLP with TensorFlow Examples

Gennaro S. Rodrigues, Ph.D.

Overview

1. Text Preprocessing
2. Sentiment Analysis with TensorFlow
3. Word Embeddings
4. Extracting Word Embeddings
5. Visualizing Word Embeddings

TensorFlow

- Open-source Machine Learning platform by Google.
- Provides tools to easily build ML models.
- Also provides other tools for NLP, such as tokenizers, datasets and pre-trained models.

Text Preprocessing

- First Step: Tokenizing

```
1   from tensorflow.keras.preprocessing.text import Tokenizer
2
3   sentences = [
4       'I love NLP',
5       'And you, do you love NLP?',
6       'You love NLP!'
7   ]
8
9   tokenizer = Tokenizer(num_words = 100)
10  tokenizer.fit_on_texts(sentences)
11  word_index = tokenizer.word_index
12  print(word_index)
```

- num_words: size of the vocabulary
- Bigger vocabulary -> more precision
  - ! More memory
  - ! More training time

Text Preprocessing

- First Step: Tokenizing

```
1   from tensorflow.keras.preprocessing.text import Tokenizer
2
3   sentences = [
4       'I love NLP',
5       'And you, do you love NLP?',
6       'You love NLP!'
7   ]
8
9   tokenizer = Tokenizer(num_words = 100)
10  tokenizer.fit_on_texts(sentences)
11  word_index = tokenizer.word_index
12  print(word_index)
```

- Creates a dictionary for the tokens (word : index)
- Like the ones we have used on previous exercises.

Text Preprocessing

- First Step: Tokenizing

```
1   from tensorflow.keras.preprocessing.text import Tokenizer
2
3   sentences = [
4       'I love NLP',
5       'And you, do you love NLP?',
6       'You love NLP!'
7   ]
8
9   tokenizer = Tokenizer(num_words = 100)
10  tokenizer.fit_on_texts(sentences)
11  word_index = tokenizer.word_index
12  print(word_index)
```

Output:

```
{
    'love': 1,
    'nlp': 2,
    ‘you': 3,
    'i': 4,
    'and': 5,
    'do’: 6
}
```

Text Preprocessing

- Second Step: Sequence

```
1   from tensorflow.keras.preprocessing.text import Tokenizer
2
3   sentences = [
4       'I love NLP',
5       'And you, do you love NLP?',
6       'You love NLP!'
7   ]
8
9   tokenizer = Tokenizer(num_words = 100)
10  tokenizer.fit_on_texts(sentences)
11  word_index = tokenizer.word_index
12  print(f'Vocabulary:{word_index}')
13  sequences = tokenizer.texts_to_sequences(sentences)
14  print(sequences)
```

- We should tokenize new texts with the same vocabulary our model was trained with (why?).
- Ideally all inputs shall have the same size (for that example).
- The tokenizer learns words from the input data.

Text Preprocessing

- Second Step: Sequence

```
1   from tensorflow.keras.preprocessing.text import Tokenizer
2
3   sentences = [
4       'I love NLP',
5       'And you, do you love NLP?',
6       'You love NLP!'
7   ]
8
9   tokenizer = Tokenizer(num_words = 100)
10  tokenizer.fit_on_texts(sentences)
11  word_index = tokenizer.word_index
12  print(f'Vocabulary:{word_index}')
13  sequences = tokenizer.texts_to_sequences(sentences)
14  print(sequences)
```

- The tokenizer learns words from the input data.
- It will fill a dictionary with all the words it finds (tokens).
- Will keep the ***num_words*** more frequent ones.

Text Preprocessing

- Second Step: Sequence

```
1   from tensorflow.keras.preprocessing.text import Tokenizer
2
3   sentences = [
4       'I love NLP',
5       'And you, do you love NLP?',
6       'You love NLP!'
7   ]
8
9   tokenizer = Tokenizer(num_words = 100)
10  tokenizer.fit_on_texts(sentences)
11  word_index = tokenizer.word_index
12  print(f'Vocabulary:{word_index}')
13  sequences = tokenizer.texts_to_sequences(sentences)
14  print(sequences)
```

Vocabulary:

```
{
    'love': 1,
    'nlp': 2,
    ‘you': 3,
    'i': 4,
    'and': 5,
    'do’: 6
}
```

Output:

```
[[4,1,2],[5,3,6,3,1,2],[3,1,2]]
```

Text Preprocessing

- First Step: Tokenizing

```
1   from tensorflow.keras.preprocessing.text import Tokenizer
2
3   sentences = [
4       'I love NLP',
5       'And you, do you love NLP?',
6       'You love NLP!'
7   ]
8
9   tokenizer = Tokenizer(num_words = 100)
10  tokenizer.fit_on_texts(sentences)
11  word_index = tokenizer.word_index
12  print(word_index)
```

Vocabulary:

```
{
    'love': 1,
    'nlp': 2,
    ‘you': 3,
    'i': 4,
    'and': 5,
    'do’: 6
}
```

Output:

```
[[4,1,2],[5,3,6,3,1,2],[3,1,2]]
```

The vocabulary is defined at the fit function. The tokenizer will NOT learn words that don’t appear in the dataset.

Text Preprocessing

- Second Step: Sequence

```
from tensorflow.keras.preprocessing.text import Tokenizer

sentences = [
    'I love NLP',
    'And you, do you love NLP?',
    'You love NLP!'
]

tokenizer = Tokenizer(num_words = 100)
tokenizer.fit_on_texts(sentences)
word_index = tokenizer.word_index
print(f'Vocabulary:{word_index}')
sequences = tokenizer.texts_to_sequences(sentences)
print(sequences)


newSent = ['I would love to learn more about machine learning and its concepts.']
newSequence = tokenizer.texts_to_sequences(newSent)
print(newSequence)
```

Vocabulary:

```
{
    'love': 1,
    'nlp': 2,
    ‘you': 3,
    'i': 4,
    'and': 5,
    'do’: 6
}
```

Output:

```
[[4,1,2],[5,3,6,3,1,2],[3,1,2]]
[[4,1,5]]
```

Text Preprocessing

- Second Step: Sequence

```
1  from tensorflow.keras.preprocessing.text import Tokenizer
2
3  sentences = [
4      'I love NLP',
5      'And you, do you love NLP?',
6      'You love NLP!'
7  ]
8
9  tokenizer = Tokenizer(num_words = 100)
10 tokenizer.fit_on_texts(sentences)
11 word_index = tokenizer.word_index
12 print(f'Vocabulary:{word_index}')
13 sequences = tokenizer.texts_to_sequences(sentences)
14 print(sequences)
15
16
17 newSent = ['I would love to learn more about machine learning and its concepts.']
18 newSequence = tokenizer.texts_to_sequences(newSent)
19 print(newSequence)
```

Vocabulary:

```
{
    'love': 1,
    'nlp': 2,
    ‘you': 3,
    'i': 4,
    'and': 5,
    'do’: 6
}
```

OOV words are ignored!

Output:

```
[[4,1,2],[5,3,6,3,1,2],[3,1,2]]
[[4,1,5]]
```

Text Preprocessing

- Second Step: Sequence, OOV

```
1  from tensorflow.keras.preprocessing.text import Tokenizer
2
3  sentences = [
4      'I love NLP',
5      'And you, do you love NLP?',
6      'You love NLP!'
7  ]
8
9  tokenizer = Tokenizer(num_words = 100, oov_token="<OOV>")
10 tokenizer.fit_on_texts(sentences)
11 word_index = tokenizer.word_index
12 print(f'Vocabulary:{word_index}')
13 sequences = tokenizer.texts_to_sequences(sentences)
14 print(sequences)
15
16
17 newSents = [
18     'I would love to learn more about machine learning and its concepts.',
19     "J'adore NLP"
20 ]
21 newSequences = tokenizer.texts_to_sequences(newSents)
22 print(newSequences)
```

We can define an OOV token to handle that.

Vocabulary:

```
{
    '<OOV>': 1,
    'love': 2,
    'nlp': 3,
    'you': 4,
    'i': 5,
    'and': 6,
    'do’: 7
}
```

Output:

```
[[5,2,3],[6,4,7,4,2,3],[4,2,3]]
[[5,1,2,1,1,1,1,1,1,6,1,1],[1,3]]
```

Text Preprocessing

- Second Step: Sequence, OOV and padding

```python
import tensorflow as tf
from tensorflow import keras

from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

sentences = [
    'I love NLP',
    'And you, do you love NLP?',
    'You love NLP!'
]

tokenizer = Tokenizer(num_words = 100, oov_token="<OOV>")
tokenizer.fit_on_texts(sentences)
word_index = tokenizer.word_index
print(f'Vocabulary:{word_index}')
sequences = tokenizer.texts_to_sequences(sentences)
print(f'Sequences:{sequences}')
paddedSequences = pad_sequences(sequences, maxlen=5)
print(f'Padded sequences:{paddedSequences}')

newSents = [
    'I would love to learn more about machine learning and its concepts.',
    "J'adore NLP"
]
newSequences = tokenizer.texts_to_sequences(newSents)
print(f'New sequences:{newSequences}')
paddedNewSequences = pad_sequences(newSequences, maxlen=5)
print(f'Padded new sequences:{paddedNewSequences}')
```

Vocabulary:

```
{
    '<OOV>': 1,
    'love': 2,
    'nlp': 3,
    'you': 4,
    'i': 5,
    'and': 6,
    'do’: 7
}
```

Output:

```
Sequences:[[5, 2, 3], [6, 4, 7, 4, 2, 3], [4, 2, 3]]
Padded sequences:[[0 0 5 2 3]
 [4 7 4 2 3]
 [0 0 4 2 3]]
New sequences:[[5, 1, 2, 1, 1, 1, 1, 1, 1, 6, 1, 1], [1, 3]]
Padded new sequences:[[1 1 6 1 1]
 [0 0 0 1 3]]
```

Sentiment Analysis: Sarcasm Detection

- https://www.kaggle.com/rmisra/news-headlines-dataset-for-sarcasm-detection


```
"root" : { 3 items
  "article_link" : string "https://www.huffingtonpost.com/entry/versace-black-code_us_5861fbefe4b0de3a08f600d5"
  "headline" : string "former versace store clerk sues over secret 'black code' for minority shoppers"
  "is_sarcastic" : int 0
}
```

Sentiment Analysis: Sarcasm Detection

- https://www.kaggle.com/rmisra/news-headlines-dataset-for-sarcasm-detection

```
df = pd.read_json("Sarcasm_Headlines_Dataset_v2.json", lines=True)
df.head()
```

	is_sarcastic	headline	article_link
0	1	thirtysomething scientists unveil doomsday clo...	https://www.theonion.com/thirtysomething-scien...
1	0	dem rep. totally nails why congress is falling...	https://www.huffingtonpost.com/entry/donna-edw...
2	0	eat your veggies: 9 deliciously different recipes	https://www.huffingtonpost.com/entry/eat-your-...
3	1	inclement weather prevents liar from getting t...	https://local.theonion.com/inclement-weather-p...
4	1	mother comes pretty close to using word 'strea...	https://www.theonion.com/mother-comes-pretty-c...

Sentiment Analysis: Sarcasm Detection

```
1   import tensorflow as tf
2   from tensorflow import keras
3   from tensorflow.keras.preprocessing.text import Tokenizer
4   from tensorflow.keras.preprocessing.sequence import pad_sequences
5   import pandas as pd
6
7   df = pd.read_json('Sarcasm_Headlines_Dataset_v2.json', lines=True)
8   print('Dataframe head:')
9   print(df.head())
10
11  tokenizer = Tokenizer(oov_token="<OOV>")
12  tokenizer.fit_on_texts(df['headline'])
13
14  sequences = tokenizer.texts_to_sequences(df['headline'])
15  paddedSequences = pad_sequences(sequences, padding='post')
16  print(paddedSequences[0])
17  print(paddedSequences.shape)
```

- Not setting the number of words in the vocabulary.
- The tokenizer will take as much as it needs (from the input data).

Sentiment Analysis: Sarcasm Detection

```
1   import tensorflow as tf
2   from tensorflow import keras
3   from tensorflow.keras.preprocessing.text import Tokenizer
4   from tensorflow.keras.preprocessing.sequence import pad_sequences
5   import pandas as pd
6
7   df = pd.read_json('Sarcasm_Headlines_Dataset_v2.json', lines=True)
8   print('Dataframe head:')
9   print(df.head())
10
11  tokenizer = Tokenizer(oov_token="<OOV>")
12  tokenizer.fit_on_texts(df['headline'])
13
14  sequences = tokenizer.texts_to_sequences(df['headline'])
15  paddedSequences = pad_sequences(sequences, padding='post')
16  print(paddedSequences[0])
17  print(paddedSequences.shape)
```

- padding=’post’ pads to the end of the strings (add/remove).
- Printing an example of padded sentence and the shape of the whole data.

Sentiment Analysis: Sarcasm Detection

```python
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

```
Dataframe head:
   is_sarcastic                                           headline                                       article_link
0             1  thirtysomething scientists unveil doomsday clo...  https://www.theonion.com/thirtysomething-scien...
1             0  dem rep. totally nails why congress is falling...  https://www.huffingtonpost.com/entry/donna-edw...
2             0  eat your veggies: 9 deliciously different recipes  https://www.huffingtonpost.com/entry/eat-your-...
3             1  inclement weather prevents liar from getting t...  https://local.theonion.com/inclement-weather-p...
4             1  mother comes pretty close to using word 'strea...  https://www.theonion.com/mother-comes-pretty-c...
[16004   355  3167  7474  2644     3   661  1119     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0]
(28619, 152)
```

- padding='post' pads to … of the strings … nove).
- … an example of … sentence and …

- 28619 texts.
- 152 words on each of them.
  - Padded to the size of the biggest text on the database.

Sentiment Analysis: Sarcasm Detection

- Stopwords don’t add a lot to the meaning of the text.
- They shall be removed

```
Dataframe head:
   is_sarcastic                                           headline                                       article_link
0             1  thirtysomething scientists unveil doomsday clo...  https://www.theonion.com/thirtysomething-scien...
1             0  dem rep. totally nails why congress is falling...  https://www.huffingtonpost.com/entry/donna-edw...
2             0  eat your veggies: 9 deliciously different recipes  https://www.huffingtonpost.com/entry/eat-your-...
3             1  inclement weather prevents liar from getting t...  https://local.theonion.com/inclement-weather-p...
4             1  mother comes pretty close to using word 'strea...  https://www.theonion.com/mother-comes-pretty-c...
[16004   355  3167  7474  2644     3   661  1119     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0]
(28619, 152)
```

Sentiment Analysis: Sarcasm Detection

- Stopwords don’t add a lot to the meaning of the text.
- They shall be removed

```
Dataframe head without stopwords:
   is_sarcastic                                           headline                                        article_link
0             1  thirtysomething scientists unveil doomsday clo...  https://www.theonion.com/thirtysomething-scien...
1             0  dem rep. totally nails congress falling short ...  https://www.huffingtonpost.com/entry/donna-edw...
2             0       eat veggies: 9 deliciously different recipes  https://www.huffingtonpost.com/entry/eat-your-...
3             1       inclement weather prevents liar getting work  https://local.theonion.com/inclement-weather-p...
4             1  mother comes pretty close using word 'streamin...  https://www.theonion.com/mother-comes-pretty-c...
[15929   268  3066  7378  2542   560  1019     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0]
(28619, 108)
```

Sentiment Analysis: Sarcasm Detection

```
stopwords = [ "a", "about", "above", "after", "again", "against", "all",

def remove_stopwords(text):
    final_text = []
    for i in text.split():
        if i.strip().lower() not in stopwords:
            final_text.append(i.strip())
    return " ".join(final_text)


df = pd.read_json('Sarcasm_Headlines_Dataset_v2.json', lines=True)
print('Dataframe head:')
print(df.head())

df['headline'] = df['headline'].apply(remove_stopwords)
tokenizer = Tokenizer(oov_token="<OOV>")
tokenizer.fit_on_texts(df['headline'])
print('Dataframe head without stopwords:')
print(df.head())
```

Embeddings in TensorFlow/Keras

- Jump to Jupyter Notebook
- Sarcasm_example.ipynb

Embeddings in TensorFlow/Keras

- Remember Embeddings? Let's see them in practice.
- Let us use embeddings to classify texts.
- Keras provides example datasets.
- https://www.tensorflow.org/datasets/catalog/overview
- In this example, let's use a simple one: the IMDb dataset for movie reviews.
  - Binary: each example is labeled as **positive** or **negative**.
  - 25000 examples for training.
  - 25000 examples for testing.

Embeddings in TensorFlow/Keras

- Jump to Jupyter Notebook
- IMDb_example.ipynb
- Visualization at https://projector.tensorflow.org
- https://www.tensorflow.org/tensorboard/tensorboard_projector_plugin

Exercise

- Try a harder sentiment analysis: goemotions.
  - 58000 Reddit comments .
  - 43410 for training.
  - 5427 for testing.
  - 5427 for validation.
  - 27 emotions categories or **neutral**.
  - https://www.tensorflow.org/datasets/catalog/goemotions
- Plot the embeddings and the learning curves. Explain them.
- Can you improve the sarcasm sentiment extraction? With a different model or more preprocessing? Visualize the word vectors (tip: sphereize and see what happens).

Exercise

- The way we are extracting the embeddings here is a little different from CBOW…
  - CBOW tried to predict a word.
  - Here, we are predicting a sentiment.
  - ***The embeddings will be related to that sentiment.***
- Can you implement the original CBOW with TensorFlow now?

Remember

- Bring up your questions!

Thank you!