|| ॐ श्री गणेशाय नमः ||

Convert Sentences into Data

Tokenization

In [1]:
```
sentence = [
    'I, love my India',
    'i love my country',
    'India is my country!'
]
```

In [2]:
```
from tensorflow.keras.preprocessing.text import Tokenizer
tokenizer = Tokenizer(num_words = 100) #Word limit : 100, 200, 10000 words.....
```

In [3]:
```
tokenizer.fit_on_texts(sentence)      #Fit the sentence
word_index = tokenizer.word_index      #check index of word in the sentennce
word_index
```

Out[3]:
```
{'my': 1, 'i': 2, 'love': 3, 'india': 4, 'country': 5, 'is': 6}
```

Check word Sequence in a sentence

In [4]:
```
sequences = tokenizer.texts_to_sequences(sentence)
sequences
```

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

In [5]:
```
# Sentences according to the word_index order
from tensorflow.keras.preprocessing.sequence import pad_sequences
padded = pad_sequences(sequences, maxlen=5, padding='pre', truncating='pre')
'''
maxlen : maximum length of each sentence,
padding : 'pre' or 'post', pad either before or after each sequence.
truncating : 'pre' or 'post', remove values from sequences larger
than `maxlen`, either at the beginning or at the end of the sequences.
```

```
'''
padded
```

Out[5]:
```
array([[0, 2, 3, 1, 4],
       [0, 2, 3, 1, 5],
       [0, 4, 6, 1, 5]])
```

In [6]:
```
padded = pad_sequences(sequences, maxlen=5, padding='post')
padded
```

Out[6]:
```
array([[2, 3, 1, 4, 0],
       [2, 3, 1, 5, 0],
       [4, 6, 1, 5, 0]])
```

In [7]:
```
padded = pad_sequences(sequences, maxlen=5, padding='pre')
```

Let's test with new sentences

In [8]:
```
test_sentence = [
    'I also love my country',
    'my country loves India also '
]
```

In [9]:
```
#Convert the text value to categorical numeric sequence
test_seq = tokenizer.texts_to_sequences(test_sentence)
test_seq
```

Out[9]:
```
[[2, 3, 1, 5], [1, 5, 4]]
```

In [10]:
```
#Use the test_seq & Let's Pad the sentence sequence
padded = pad_sequences(test_seq, maxlen=5)
print("Padded Sequences:")
print(padded)

print("\nWord Index = " , word_index)
```

```
Padded Sequences:
[[0 2 3 1 5]
 [0 0 1 5 4]]

Word Index =  {'my': 1, 'i': 2, 'love': 3, 'india': 4, 'country': 5, 'is': 6}
```

As we can see from above that our test sentence is :

```
'I also love my country',
'my country loves India also '
```

& if we match the Padded sequences with word index.. We will find that it is ignoring the new word (Here :'also')

- **Let's tokenize the sentence with< oov_token >. This will provide an index to undefined or new words.**

In [11]:
```
sentences = [
    'I, love my India',
    'i love my country',
    'India is my country!',
    'What is your country name?'
]
```

In [12]:
```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

tokenizer = Tokenizer(num_words = 100, oov_token="<OOV>")
tokenizer.fit_on_texts(sentences)
word_index = tokenizer.word_index

sequences = tokenizer.texts_to_sequences(sentences)

padded = pad_sequences(sequences, maxlen=5, padding ='pre', truncating='post')
print("sentences = " ,sentences)
print("\nWord Index = " , word_index)
print("\nSequences = " , sequences)
print("\nPadded Sequences:")
print(padded)
```

```
sentences =  ['I, love my India', 'i love my country', 'India is my country!', 'What is your country name?']

Word Index =  {'<OOV>': 1, 'my': 2, 'country': 3, 'i': 4, 'love': 5, 'india': 6, 'is': 7, 'what': 8, 'your': 9, 'name': 10}

Sequences =  [[4, 5, 2, 6], [4, 5, 2, 3], [6, 7, 2, 3], [8, 7, 9, 3, 10]]

Padded Sequences:
[[ 0  4  5  2  6]
 [ 0  4  5  2  3]
 [ 0  6  7  2  3]
 [ 8  7  9  3 10]]
```

In [13]:

```
# Try with words that the tokenizer wasn't fit to
test_sentence = [
    'I also love my country',
    'I belong from India',
    'The capital of India is Delhi'
]
print("test_sentence = ", test_sentence)

print("\nWord Index = " , word_index)

test_seq = tokenizer.texts_to_sequences(test_sentence)
print("\nTest Sequence = ", test_seq)

padded = pad_sequences(test_seq, maxlen=10)
print("\nPadded Test Sequence: ")
print(padded)
```

```
test_sentence =  ['I also love my country', 'I belong from India', 'The capital of India is Delhi']

Word Index =  {'<OOV>': 1, 'my': 2, 'country': 3, 'i': 4, 'love': 5, 'india': 6, 'is': 7, 'what': 8, 'your': 9, 'name': 10}

Test Sequence =  [[4, 1, 5, 2, 3], [4, 1, 1, 6], [1, 1, 1, 6, 7, 1]]

Padded Test Sequence:
[[0 0 0 0 0 4 1 5 2 3]
 [0 0 0 0 0 0 4 1 1 6]
 [0 0 0 0 1 1 1 6 7 1]]
```

As we can see from above that tokenizer is not ignoring any word. It is just assigning the new word to word_index : 1

NLTK : Natural Language Toolkit

In [14]:
```
pip --version
```

```
pip 22.1.1 from E:\Python_3.10\lib\site-packages\pip (python 3.10)

Note: you may need to restart the kernel to use updated packages.
```

In [15]:
```
#pip install pipenv
```

In [16]:
```
#cd project_folder
#pipenv install requests
```

In [17]:
```
import requests
```

In [18]:
```
import nltk
```

In [19]:
```
#nltk.download()
```

In [20]:
```
from nltk.corpus import brown
brown.words()
```

Out[20]:
```
['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]
```

In [21]:
```
sentence = """At eight o'clock on Thursday morning Arthur didn't feel very good."""
```

In [22]:
```
tokens = nltk.word_tokenize(sentence)
```

In [23]:
```
tokens
```

Out[23]:
```
['At',
 'eight',
 "o'clock",
 'on',
 'Thursday',
 'morning',
 'Arthur',
 'did',
 "n't",
 'feel',
 'very',
 'good',
 '.']
```

In [24]:
```
tagged = nltk.pos_tag(tokens)
tagged
```

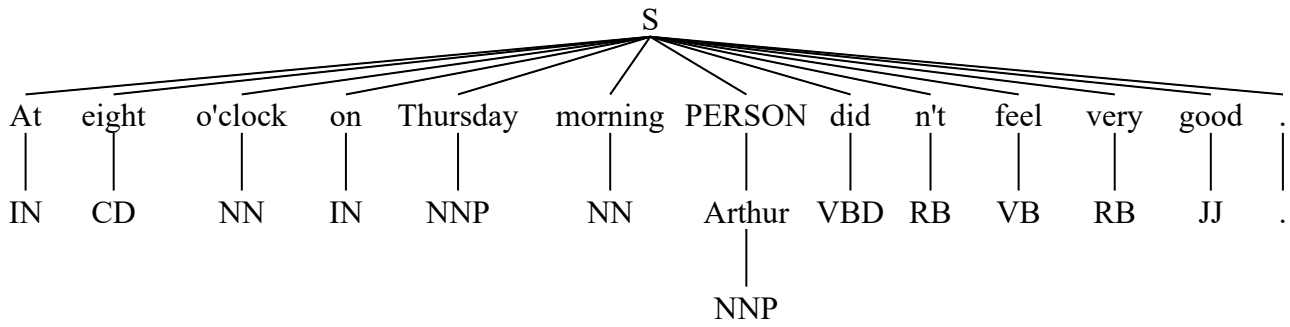
Out[24]:
```
[('At', 'IN'),
 ('eight', 'CD'),
 ("o'clock", 'NN'),
 ('on', 'IN'),
 ('Thursday', 'NNP'),
 ('morning', 'NN'),
 ('Arthur', 'NNP'),
 ('did', 'VBD'),
 ("n't", 'RB'),
 ('feel', 'VB'),
 ('very', 'RB'),
 ('good', 'JJ'),
 ('.', '.')]
```

In [25]:
```
tagged[0:6]
```

Out[25]:
```
[('At', 'IN'),
 ('eight', 'CD'),
 ("o'clock", 'NN'),
 ('on', 'IN'),
 ('Thursday', 'NNP'),
 ('morning', 'NN')]
```

In [26]:
```
entities = nltk.chunk.ne_chunk(tagged)
entities
```

Out[26]:



In [27]:
```
from nltk.corpus import treebank
t = treebank.parsed_sents('wsj_0001.mrg')[0]
t.draw()
```

In [ ]:
```

```