11. NLP - Basic Text Classification [GloVe, Cosine]

```
## pip install glove-python3
import re
import numpy as np
from glove import Corpus, Glove

## Dataset
dataset = [
    "Tears fell, heartache lingered silently.",
    "His absence echoed in emptiness.",
    "Rain matched her somber mood.",
    "Silence enveloped his lonely heart.",
    "Sorrow weighed heavily on him.",
    "Their eyes met, hearts connected.",
    "True love lasts a lifetime.",
    "Every moment together felt magical.",
    "Love brings joy and peace.",
    "Embracing, they felt infinite love.",
    "His fists clenched with rage.",
    "Anger burned in her eyes.",
    "Voices rose, argument escalated quickly.",
    "Slammed door echoed her fury.",
    "Frustration fueled their heated exchange."
]
targets = ["Sadness", "Sadness", "Sadness", "Sadness", "Sadness", "Love", 
"Love", "Love", "Love", "Love", "Anger", "Anger", "Anger", "Anger", "Anger"]
print(f'Dataset Length : {dataset.__len__()}')
```

```
Dataset length : 15
```

```
## Preprocess dataset: Tokenization and cleaning
def preprocess_text(text):
    ## Lowercase the text
    text = text.lower()
    ## Remove punctuation and special characters
    text = re.sub(r'[^a-z\s]', '', text)
    ## Tokenize by splitting on spaces
    tokens = text.split()
    return tokens

## Tokenize the dataset
tokenized_data = [preprocess_text(sentence) for sentence in dataset]
print(f'Preprocessing : \n {tokenized_data}')
```

```
Preprocessing :
[['tears', 'fell', 'heartache', 'lingered', 'silently'],
 ['his', 'absence', 'echoed', 'in', 'emptiness'],
 ['rain', 'matched', 'her', 'somber', 'mood'],
 ['silence', 'enveloped', 'his', 'lonely', 'heart'],
 ['sorrow', 'weighed', 'heavily', 'on', 'him'],
 ['their', 'eyes', 'met', 'hearts', 'connected'],
 ['true', 'love', 'lasts', 'a', 'lifetime'],
 ['every', 'moment', 'together', 'felt', 'magical'],
 ['love', 'brings', 'joy', 'and', 'peace'],
 ['embracing', 'they', 'felt', 'infinite', 'love'],
 ['his', 'fists', 'clenched', 'with', 'rage'],
 ['anger', 'burned', 'in', 'her', 'eyes'],
 ['voices', 'rose', 'argument', 'escalated', 'quickly'],
 ['slammed', 'door', 'echoed', 'her', 'fury'],
 ['frustration', 'fueled', 'their', 'heated', 'exchange']]
```

```python
## Build the co-occurrence matrix using Corpus in the GloVe library
corpus = Corpus()
corpus.fit(tokenized_data, window=2)
print(f'Co-occurrence Matrix : \n{corpus.matrix.data}')
```

```
Co-occurrence Matrix :
[1.  0.5 1.  0.5 1.  0.5 1.  1.  0.5 0.5 1.  1.  0.5 1.  0.5 1.  0.5 1.
 0.5 1.  0.5 1.  0.5 1.  1.  0.5 0.5 1.  1.  0.5 1.  0.5 1.  0.5 1.  0.5
 0.5 1.  1.  1.  0.5 1.  1.  0.5 1.  0.5 1.  0.5 1.  1.  0.5 0.5 1.  1.
 0.5 1.  0.5 1.  0.5 1.  1.  0.5 1.  0.5 0.5 1.  0.5 1.  1.  0.5 1.  1.
 0.5 1.  0.5 1.  0.5 1.  0.5 1.  1.  1.  0.5 1.  0.5 1.  1.  0.5 1.  0.5
 1.  0.5 1.  1.  1.  0.5 1.  0.5 1.  0.5 1.  1.  1.  0.5 1. ]
```

```python
## Train Custom GloVe model
my_glove = Glove(no_components=50, learning_rate=0.05)
my_glove.fit(corpus.matrix, epochs=30, no_threads=4, verbose=False)
my_glove.add_dictionary(corpus.dictionary)

## Save the model
my_glove.save("custom_glove.model")
```

```
Output : custom_glove.model
```

```python
print(f'GloVe Dictionary : {my_glove.dictionary}')
```

```
GloVe Dictionary : {'tears': 0, 'fell': 1, 'heartache': 2, 'lingered': 3,
'silently': 4, 'his': 5, 'absence': 6, 'echoed': 7, 'in': 8, 'emptiness': 9,
'rain': 10, 'matched': 11, 'her': 12, 'somber': 13, 'mood': 14, 'silence': 15,
'enveloped': 16, 'lonely': 17, 'heart': 18, 'sorrow': 19, 'weighed': 20,
'heavily': 21, 'on': 22, 'him': 23, 'their': 24, 'eyes': 25, 'met': 26,
'hearts': 27, 'connected': 28, 'true': 29, 'love': 30, 'lasts': 31, 'a': 32,
'lifetime': 33, 'every': 34, 'moment': 35, 'together': 36, 'felt': 37,
'magical': 38, 'brings': 39, 'joy': 40, 'and': 41, 'peace': 42, 'embracing':
43, 'they': 44, 'infinite': 45, 'fists': 46, 'clenched': 47, 'with': 48,
'rage': 49, 'anger': 50, 'burned': 51, 'voices': 52, 'rose': 53, 'argument':
54, 'escalated': 55, 'quickly': 56, 'slammed': 57, 'door': 58, 'fury': 59,
'frustration': 60, 'fueled': 61, 'heated': 62, 'exchange': 63}
```

```python
## Load the GloVe model (for inference)
custom_glove = Glove.load("custom_glove.model")

## To convert a sentence into a vector by averaging GloVe word vectors
def sentence_to_vector(sentence, glove, vector_size=50):
    tokens = preprocess_text(sentence)
    vectors = [glove.word_vectors[glove.dictionary[word]] for word in tokens
if word in glove.dictionary]

    ## If no word in the sentence is in the model, return a zero vector
    if len(vectors) == 0:
        return np.zeros(vector_size)
    ## Average of all word vectors
    return np.mean(vectors, axis=0)

## Convert the entire dataset into sentence vectors
features = np.array([sentence_to_vector(sentence, custom_glove,
vector_size=50) for sentence in dataset])
print(f"Sentence Embedding of '{dataset[0]}' is \n {features[0]}")
```

```
Sentence Embedding of 'Tears fell, heartache lingered silently.' is
 [ 1.01248410e-04  2.97859958e-03 -8.49863411e-04  7.89478623e-04
  6.32850062e-05  1.05415465e-03  2.51425468e-03  8.11888875e-04
 -1.86157104e-03 -1.87267965e-03  2.08153137e-03  2.66716264e-04
  5.36957895e-03  5.01229314e-03 -2.36265281e-04 -4.14386249e-03
  9.45040670e-04 -1.56356608e-04  2.65767263e-03 -5.23199170e-04
  4.72795615e-04 -1.09920075e-03 -1.54849645e-03 -2.06304279e-03
  1.76821259e-03 -2.67534115e-03 -8.45719415e-04  2.89620970e-04
  9.08952261e-04 -4.42916767e-04  2.63091922e-04  2.53748545e-03
  2.19742259e-03 -5.74414927e-04  2.95407257e-03 -8.63746531e-04
 -2.33961102e-03  2.94672111e-03 -3.29882915e-04 -7.32885847e-04
  4.94018994e-03  1.06338458e-03 -1.46593202e-03 -2.03628287e-03
 -2.04029811e-03 -2.42719847e-04  1.01037211e-04  2.81178645e-03
 -8.05944890e-04 -1.93777324e-03]
```

```
## Classifier
def cosine_similarity(A, B):
    dot_product = np.dot(A, B)
    norm_A = np.linalg.norm(A)
    norm_B = np.linalg.norm(B)
    return dot_product / (norm_A * norm_B)
## Test cosine similarity between two sentences
similarity = cosine_similarity(features[1], features[2])
print("Cosine Similarity:", similarity)
```

```
Cosine Similarity: -0.13098348822189457
```

```
## Inference
test = "Real love endures forever."
test_vector = sentence_to_vector(test, custom_glove)
preds = [cosine_similarity(source, test_vector) for source in features]
pred = targets[np.argmax(preds)]
print("Prediction:", pred)
```

```
Prediction : Love
```

Ko Yin Maung

Koyinmaung007@gmail.com