# **Text Summarization with Encoder Decoder LSTM Networks**

# **Read Me**

## **Pre-Processing**

Python File: CAP\_PreProc\_Reviews.py

Steps:

1. Load Data
2. Prep Data: replace contractions, remove special characters & stop words
3. Dataset Analysis
   * Tokenize Reviews and Summaries with Keras
   * Load GloVe Word Embeddings
   * Check counts, word frequency and document frequency
4. Tokenize with Function
   * Create dictionary to convert words to integers.
     + Limit the vocabulary words to words that have frequency > a threshold
     + Add special tokens to vocabulary
   * Create a dictionary to convert integers back to words
   * Create matrix with Glove Embeddings for word integers in vocabulary
5. Texts to Sequence
   * Convert words in dataset Reviews and Summaries to integers
   * Count unique words in Reviews & Summaries
   * Count number of words that are not found in GloVe embedding
6. Sentence Lengths
   * Get sentence lengths for all Reviews and Summaries
   * Analyze sentence lengths and decide max sentence length for Reviews and Summaries
   * Create Reduced dataset that only includes Reviews and Summaries that meet specific conditions
     + Limit the length of summaries and texts based on the min and max ranges
     + Remove reviews that include too many UNKs
7. Subset & Pad Dataset for Training
   * Get subset of already reduced dataset with specific length of words in sentences
   * Pad sequences of index to have same length
8. Turn reduced Text and Summary Sets to back to text format
9. Format Encoder & Decoder Target Data for Model w/ untrained Embeddings
   * Reduced Reviews & Summaries: Turn texts back to integers using the global vocabulary to integer dictionary
   * Get number of unique tokens in reduced text and summary data
   * Create dictionary that maps a new integer to the word integer from the original dataset
     + This allows user to select different subsets of reviews & summaries, yet still retain the original integer the vocabulary word was mapped to.
   * Create matrices for encoder input data, decoder input data, and decoder target data
10. Store Data for Models w/ untrained Embeddings
    * Store reduced word format Reviews and Summaries to csv
    * Store Encoder and Decoder data into h5 matrices
    * Store dictionaries into pickle
11. Format Encoder & Decoder Target Data for Model w/ Trained Embedding (GloVe)
    * Use Keras Tokenizer to map an integer to a unique word from the word formatted reduced dataset
    * Turn Reviews & Summaries into token format
    * Get number of unique tokens in reduced text and summary data
12. Store data for model w/ Trained Embeddings
    * Store reduced word format Reviews and Summaries to csv
    * Store Encoder and Decoder data into h5 matrices
    * Store dictionaries into pickle
13. Store data for auto model
    * Create encoder and decoder data matrices in format required for auto model
    * Store Encoder and Decoder data into h5 matrices

## **Encoder Decoder Models with Untrained Embeddings**

Python Files:

* s2s\_model\_emb.py
  + LSTM Units = 50
  + Batch Size = 100
* s2s\_model\_emb2.py
  + LSTM Units = 25
  + Batch Size = 100
* s2s\_model\_emb3.py
  + LSTM Units = 75
  + Batch Size = 100
* s2s\_model\_emb4.py
  + LSTM Units = 100
  + Batch Size = 100
* s2s\_model\_emb4\_b50.py
  + LSTM Units = 100
  + Batch Size = 50

Steps:

1. Load Data
   1. Encoder Decoder matrices
   2. Reviews and Summaries csv
   3. Dictionary pickle files
2. Build Model
   1. Encoder Model
   2. Decoder Model
   3. Compile & Train Model
   4. Sampling Model
   5. Generate Sequences through Inference Process
      1. Generate sequence using sampling model
      2. Turn sequence integer back to word format
3. Save csv with Reviews, Summaries, and model Predicted Summaries

## **Encoder Decoder Models with GloVe Pre trained Embeddings**

Python Files:

* s2s\_model\_emb\_glove.py
  + LSTM Units = 50
  + Batch Size = 100
* s2s\_model\_emb\_glove2.py
  + LSTM Units = 25
  + Batch Size = 100
* s2s\_model\_emb\_glove3.py
  + LSTM Units = 75
  + Batch Size = 100

Steps:

1. Load Data
   1. Encoder Decoder matrices
   2. Reviews and Summaries csv
   3. Dictionary pickle files
2. GloVe Embeddings
   1. Load Pre-Trained Word Embeddings
   2. Create Embedding Matrix for Encoder and Decoder Data
3. Build Model
   1. Encoder Model
   2. Decoder Model
   3. Compile & Train Model
   4. Sampling Model
   5. Generate Sequences through Inference Process
      1. Generate sequence using sampling model
      2. Turn sequence integer back to word format
4. Save csv with Reviews, Summaries, and model Predicted Summaries

## **Encoder Decoder Models with Automated Sequence Generation**

Python Files:

* s2s\_emb\_iter.py
  + LSTM Units = 100
  + Batch Size = 100

Steps:

1. Load Data
   1. Encoder Decoder matrices
   2. Reviews and Summaries csv
   3. Dictionary pickle files
2. GloVe Embeddings
   1. Load Pre-Trained Word Embeddings
   2. Create Embedding Matrix for Encoder and Decoder Data
3. Build Model
   1. Encoder Model
   2. Decoder Model
   3. Tie Encoder and Decoder Models Together and Train
   4. Predict new Summary sequences using best model
      1. Turn sequence integer back to word format
4. Save csv with Reviews, Summaries, and model Predicted Summaries

## **Model Evaluation**

Python File: evaluate\_emb.py

Steps:

1. Read in reviews, summaries, and predictions output from models (csv)
2. Evaluate how many words from predicted text are in the summary and how many words from predicted text are in the reviews text
3. Evaluate if sentiment from summaries and predicted summaries match
   1. Remove special tokens from text (<GO>,<PAD>,<EOS>)
   2. Use text blob to determine sentiment for summaries and predictions
   3. Get score of matching sentiments