Word2vec.ipynb

Preprocessing & Helper Functions

build_data() : loads the dataset/wikihowAll.csv. The file is assumed in this location. You can change the global variable path to change directory. The wikidataset provided the code to load the dataset as a dataframe. We reused part of their code. The function returns list of articles, and list of titles.

pre_process(): this function takes a list of string as parameter and returns returns a preprocessed list. Digits, punctuations are removed. The words is converted into its lemma form and are checked against a stop word corpus.

build_tokenizer(): This function takes as an input a list and out of vocabulary token and returns a tokenizer fit on the given dataset.

get_reverse_word_index(): returns reverse word dictionary given a tokenizer.

addEOSTokens(data): adds <start> token and <eos> token.

The following block shows the preprocess pipeline (included in the file)

```
vocab_size = 50000
oov_tok = "<OOV>"
articles_raw_text, summaries_raw_text = build_data()
articles_processed = pre_process(articles_raw_text)
articles_processed = addEOSTokens(articles_processed)
```

```
tokenizer = build_tokenizer(articles_processed,oov_tok)
x_seq = tokenizer.texts_to_sequences(articles_processed)
x_data = pad_sequences(x_seq,maxlen=100)
```

Word2vec implementation

contains an encoder-decoder implementation for text summarization using word2vec embeddings.

The first step is to load the embedding matrix using loadWord2vec() function. This function takes as a parameter the dataset (articles), tokenizer, size of embedding dimension. The function loads word2vec model using gensim.models.Word2Vec. The min count used is 15 and iteration = 10.

Architecture:
The neural network is composed of an encoder and decoder.

The encoder : Embedding layer and 3 stacked LSTM layers. The input of the encoder is defined as shape=(lenth_of_the_article,)
The embedding layer dimension is set to 100. This should match the parameters defined for the word2vec model to successfully load the embedding matrix (weights). The encoder input the articles tokenized sequence.

Decoder: The decoder is composed an embedding layer, a single lstm layer , an attention layer and a dense layer wrapped in a TimeDistributed layer. The input of the decoder is the summaries tokenized sequence of length 10. The number of units in the dense layer is set to the vocabulary size. The output activation is softmax since we want a probability distribution over the vocabulary tokens as output for each time step.

The vocabulary size is set to the size of different tokens . The latent dimensions of the lstm layers is set to 100. The embedding dimension is

100. The input of the decoder embedding layer is the output of the last encoder lstm layer. The hidden state and cell state of the last encoder layer is passed to the first decoder lstm layer.

The optimizer used is rmsprop with learning rate of 0.001

Encoder Inference:
An inference encoder model is defined with the same input as the model and outputs the last lstm encoder output.
encoder_model =
Model(inputs=encoder_inputs,outputs=[encoder_outputs, state_h, state_c])

Decoder inference:
An inference decoder model is defined. The input shape of the decoder inference is similar to the main model decoder input. The input is concatenated with the output of the encoder, the encoder hidden state output and the encoder cell state output.

decode_sequence(): decode sequence function can be used to output prediction of the model. It takes a sequence as an input and outputs the decoded sequence/summary.

The main function creates an encoder and decoder and trains the model with the defined parameters. The loss is defined as sparse categorical entropy. This allow us to input the sequence to the decoder without changing the test data into one hot encoded.

The implementation of the attention based architecture is based on the following implementation : https://towardsdatascience.com/neural-machine-translation-nmt-with-attention-mechanism-5e59b57bd2ac