# Introduction

# In this work we propose a data-driven approach based on Neural Network, without relying heavily on human help. We developed a general framework for text summarization. The model was trained on [amazon food review data](https://www.kaggle.com/snap/amazon-fine-food-reviews) corpora containing thousands of reviews-summary pairs. The summarization tasks considered in this project is when given a Text consisting of a sequence of sentences. We are interested in obtaining the best abstract for summaries considering context and sentiment of the review.

# Previous solutions: there have been many solutions applied including, RNN, LSTM, bidirectional-LSTM, bidirectional-GRU.

# Dataset: amazon food reviews <https://www.kaggle.com/snap/amazon-fine-food-reviews>

# proposed method: Seq2Seq

**Figure 1 Proposed Model**

# Evaluation method:

# ROUGE: recall-based evaluation, predictions were save in doc named “”

# BLEU: precision-based evaluation metric, predictions were save in doc named “”

# Model Building:

# Cleaning

# Removing anomalies

1. **Tokenization (Text Tokenizer + Summary Tokenizer:** dividing the text into a set of meaningful pieces (tokens)
   1. **Word index**
   2. **Sequences**
   3. **Padding sequences**

# Stemming and lemmatization

# I tried both and they hurt the accuracy so I removed them both

# I believe Stemming return the word to its roots which affects the context (and sentiment) of the sentences and eventually hurt the accuracy when doing the summarization

|  |  |
| --- | --- |
| stemming with porter results were the worst | snow ball the results were slightly better but still worse than without stemming at all |
|  |  |

1. **Embedding** is a relatively low-dimensional space into which you can translate high-dimensional vectors. Embedding makes it easier to do machine learning on large inputs like sparse vectors representing words. Ideally, an embedding captures some of the semantics of the input by placing semantically similar inputs close together in the embedding space.

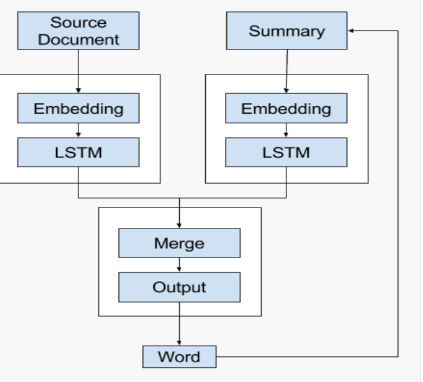
# LSTM –RNN different architectures tried out

# I tried GRU and bidirectional GRU: The model wasn’t able to run on my laptop

# I Tried bidirectional LSTM: The bidirectional LSTM took almost double the time with no actual enhancement on the accuracy of the model so I removed the bidirectional and kept the classic LSTM, below the results with bidirectional LSTM, also I noticed that the parameters number was doubled (which is expected in such case) please see below:

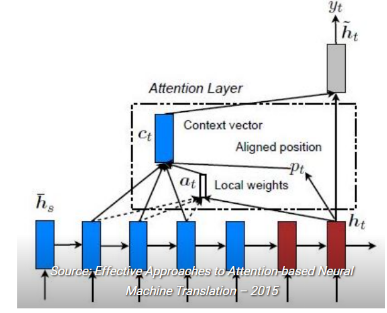
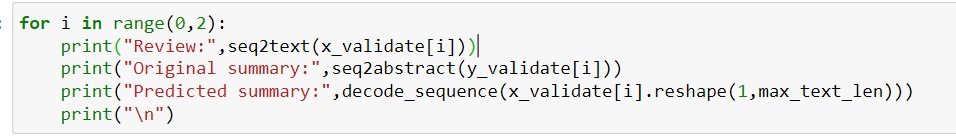
# 

### **Recursive Model**

****Here I sued the recursive model not the one shot model, where the decoder uses the context vector and the distributed representation of all words generated so far as input in order to generate the next word.

* 1. **Encoder:** An Encoder Long Short Term Memory model (LSTM) reads the entire input sequence wherein, at each time step, one word is fed into the encoder. It then processes the information at every time step and captures the contextual information present in the input sequence.
  2. **Decoder:** The decoder is also an LSTM network which reads the entire target sequence word-by-word and predicts the same sequence offset by one timestep. The decoder is trained to predict the next word in the sequence given the previous word.
  3. **Inference Phase:** After training, the model is tested on new source sequences for which the target sequence is unknown. So, we need to set up the inference architecture to decode a test sequence

**Note:** special *Start, End* token used are**: sostok, eostok**

1. **Attention Mechanism:** the encoder ,decoder mechanism works only for short sequences I sued *Local Attention* where only a few hidden states of the encoder are considered for deriving the attended context vector
2. **Prediction: below I printed 2 samples of the predicted abstracts VS the original ones**

* **Review**: simply bar simply fantastic love vegan high protein helpful need find alternative protein sources also provide amazing flavors favorite peanut butter one

**Original summary**: simply awesome

**Predicted summary**: delicious

* **Review**: longer find flavored tea stores amazon place get thanks amazon

**Original summary**: raspberry tea

**Predicted summary**: great tea

1. **How to enhance the model**

* Using **GLOVE** for embedding, it’s a way of transfer models where it will definitely enhance the word2vec embedding that I currently have.
* **Beam search**: it will find the best N number of words with best probability of each word of the title which enhances the accuracy and it includes the previous words in the following prediction yet, removes the picked word from the later ones which decreases word redundancy in the prediction.

# results and discussion

Building the above model (with embedding, 3 layers of LSTM for the encode and 1 LSTM for the decoder without bidirectional LSTM, attention layer) gave good results and a good model with good F1. However, I am eager to enhance the model to reach better accuracy/Recall. It is worth noting here that changing the optimization parameters gave different results while building the model; so it’s worth trying to hyper-parameter optimization and Gaussian Process for better results. Also, C-RNN-GAN looks like a promising model to be tested for this problem.