# Text Summarization

**Devansh Sundeep Mody**
Department of Computer Science
Lakehead University
dmody@lakeheadu.ca

## Abstract

Text summarization is the process of creating short summaries from reviews, documents or any other longer forms of text. In this project an abstractive text summarizer is built using Long Short Term Memory (LSTM) based seq2seq encoder decoder with attention mechanism and the model is compared with T5-small transformer model for the reviews of products sold on Amazon website. Abstractive summarization preserves the meaning and generates more human like summaries. Abstractive methods select words based on semantic understanding, relation between words and how they are structured and generate summary. They also require deeper understanding of the text. There were few challenges like modeling key words, sequence to sequence word structure and emitting words that are rare or unseen at training time which are solved by using attention based encoder and decoder architecture with LSTM. There are wide range of application areas of text summarization like media monitoring, newsletters, social media marketing and others where a short summary is required.

## 1 Introduction

Websites such as Amazon allow customers to leave reviews for various products. There are usually hundreds of reviews for a single product each review could be lengthy and repetitive. Therefore automatic review summarization has a huge potential in that it could help customers to make quick decisions on certain products.

Automatic summarization is an active field in Natural Language Processing. There are two types of summarization extractive and abstractive.

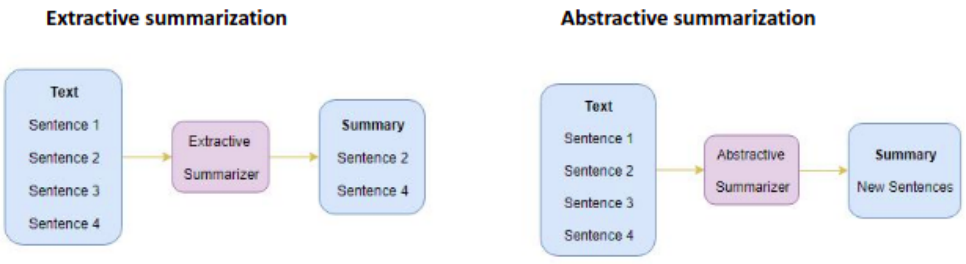


Figure 1: Extractive and Abstractive Text Summarization

Extractive methods attempt to summarize articles by selecting a subset of words that retain the most important points, while abstractive methods select words based on semantic understanding, even those words did not appear in the source documents. Although much work has been done in the area of extractive summarization, limited study is available for abstractive summarization.

Data set used for the project is downloaded from the link `https://nijianmo.github.io/amazon/index.html` The reviews data set of shoes, clothing and jewellery will be used, this data set contains multiple columns out of which only review and summary column will be used. The size of the data set is 6gb and has more than 2,70,000

## 2 Background/Related Work

There have been several papers studying the topic of automatic summarization over the years, most of them focus on extractive summarization. The very first work is published by Luhn [1] in the 1950s. He proposed a simple idea that some words in a document are more descriptive of its content than others. He also suggested using word counts to find those words, because words that occur often are likely to be the main topic of the document. Of course we now know that words with higher frequency are not necessarily the main topic and thus his approach cannot work perfectly well, but his work shapes much of later research.

Later research refines the idea of using raw frequency of words by using TF*IDF weights [2] and log likelihood ratio test for topic signatures [3]. In contrast, there has been limited study in abstractive summarization. There have been several papers in machine translation, such as recurrent neural network encoder decoder proposed by Cho et al. [4] and an extension of RNN encoder decoder with attention mechanism by Bahdanau et al. [5]. Recently Many model using the transformer based architecture [6] and Multi head attention with pointer mechanism are proposed[7] and all these models give a better performance. The evaluation metrics like bleau and rouge doesn't consider synonyms so even if the summarizer generates a synonym of original summary, metrics will give them a score of zero. So evaluation criteria needs to be updated for the efficient evaluation of the corpus.

## 3 Approach

In this section the detailed architecture of the models used, data pre processing, evaluation metrics used along with the mathematical formulas will be described.

### 3.1 Problem Statement

Given a piece of review, text summarizer generates a shorter version of the review while preserving the sentiment and points. More formally, Let the input sequence of M words be x 1 , ..., x M coming from a vocabulary of size V. To generate a shortened sequence y 1 , ..., y N such that N < M and y retains the essence of x. An assumption is made that all y 1 , ...y N are coming from the same vocabulary.

### 3.2 Data set and Data Preparation

The data set of clothing, shoes and jwellery reviews is downloaded as a reviews_Clothing_shoes_and_Jwelry_5.json.gz file. Size of the data set is 6GB and contains more than 2,70,000 rows of review, summary and other columns out of which only review and summary column will be used. As the size of the data is still to large so only 100000 to 180000 rows are used and data is randomly split into 70percent as training and 15 percent as testing and validation set.

A lot of work and effort is put on data pre processing to make sure the data feed to the model is accurate enough for the model to generate meaningful and human like similar summary.

### 3.3 Data pre processing and preparing data for model

- Stop words are downloaded from three library nltk, spacy and gensim and one combined final list of stop words is created.
- Contractions were used from the link `https://stackoverflow.com/questions/19790188/expanding-english-language-contractions-in-python`.
- Word embeddings were downloaded from the link `https://conceptnet.s3.amazonaws.com/downloads/2019/numberbatch/numberbatch-en-19.08.txt.gz`
- **Following are detailed steps involved in data pre processing**
  - The input text is converted to lower case.
  - Text is split into words and words are replaced with their contractions. Replacement of words with their longer forms is required to have a uniformity and to get accurate results. The new addition by me to the list of existing contractions is "rec'd": "received"
  - Unwanted words like https, hyperlinks, trailing spaces, slashes are removed. string.printable method is used it contains a list of punctuation's, ASCII characters, digits and white spaces so all these things are removed from the text.
  - Remove stop words
  - Finally cleaned data is written into product_reviews.csv file
- **Create vocabulary and word embedding matrix**
  - Read the conceptual number batch word embedding file numberbatch-en-19.08.txt
  - Tokenize words and create a vocabulary
  - Replace text with their token number which is the index position of word in the vocabulary
  - Add special tokens UNK, EOS, PAD and GO at the end of the vocabulary
  - If a word exists in word embedding matrix then get word embedding for the given word or initialize it with random number and generate word embedding matrix.
- **once vocabulary, word embedding matrix is generated the data pre processing step is complete we have our word embeddings and our vocabulary ready.**
- **Additional filter are applied on data to further improve the quality of data that goes to model.**
  - Pad the sentences to the given maximum length for both review and summary columns
  - The limit for UNK token in review is set to one and for summary is set to zero so if number of UNK is greater than one for a given review or greater than zero for a given summary drop that particular row only
  - Minimum review length is 2 and maximum review length is 80 so if review length exceeds the limit or its less than the minimum review limit drop the row
  - Maximum summary length is set to 10 so if the length of summary is greater than maximum length drop the row
- **After applying all the steps we get our final refined data which is feed to our model**

I have test my model for different lengths of data so here below is snapshot of data after pre processing. Snapshot1 is the amount of data used when i was testing the model when it was full functional and there were no errors in code. Snapshot2 is the data used for my final testing to get the results the 1130532_textsummarynlpproject.ipynb file submited with project that was tested on data based on the image of snapshot2.

```
The length of dataset is  180000
vocab length 68861
Word embeddings: 516783
Number of words missing from word_embeddings: 728
Percent of words that are missing from our
vocabulary: 1.06%
Total number of unique words: 68861
Number of words we will use: 37429
Percent of words we will use: 54.35%
length vocab_to_int 37429
length int_to_vocab 37429
length of word embedding matrix 37429
after word to index for reviewText [0, 3912, 0, 17,
12, 119, 278, 209, 79, 905, 3912, 1533]
after word to index for summary [37428, 0, 3912,
70, 1154, 565, 37427]
total number of unk token are 0
length of dataset that will be used 168285
length of split datasets train 117799, test 25243
and validation 25243
Vocabulary Size: 37429
voc_to_int_ 37425 37426 37427
```

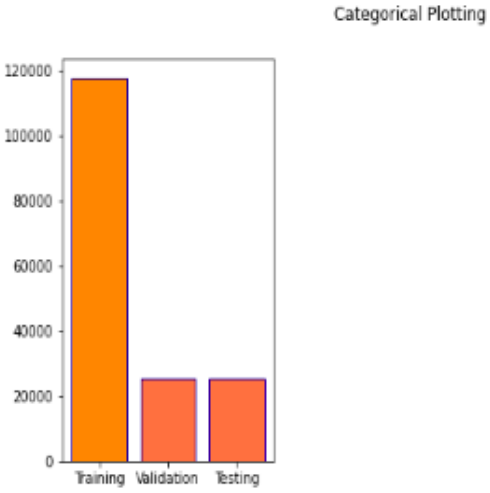


Figure 2: Snapshot1 of data used for model

```
The length of dataset is  180000
vocab length 68861
Word embeddings: 516783
Number of words missing from word_embeddings: 728
Percent of words that are missing from our vocabulary:
1.06%
Total number of unique words: 68861
Number of words we will use: 37429
Percent of words we will use: 54.35%
length vocab_to_int 37429
length int_to_vocab 37429
length of word embedding matrix 37429
after word to index for reviewText [0, 3920, 0, 17, 12,
119, 278, 209, 79, 905, 3920, 1532]
after word to index for summary [37428, 0, 3920, 70,
1154, 565, 37427]
total number of unk token are 0
length of dataset that will be used 162996
length of split datasets train 114097, test 24449 and
validation 24450
Vocabulary Size: 37429
voc_to_int_ 37425 37426 37427
```

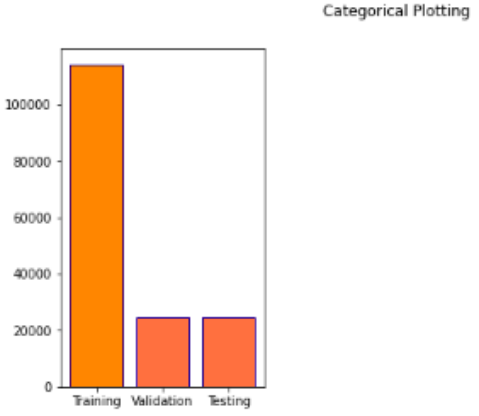


Figure 3: Snapshot2 of data used for model

### 3.4 Model design

- Long sort term memory (LSTM) based encoder decoder model with attention mechanism is used and its performance is compared with T5-small transformer model.

- **LSTM Model**
  - LSTM based seq2seq encoder decoder model with attention used in our project

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| input_1 (InputLayer) | [(None, 80)] | 0 | |
| embedding (Embedding) | (None, 80, 300) | 11228700 | input_1[0][0] |
| lstm (LSTM) | [(None, 80, 80), (No | 121920 | embedding[0][0] |
| input_2 (InputLayer) | [(None, None)] | 0 | |
| lstm_1 (LSTM) | [(None, 80, 80), (No | 51520 | lstm[0][0] |
| embedding_1 (Embedding) | (None, None, 300) | 11228700 | input_2[0][0] |
| lstm_2 (LSTM) | [(None, 80, 80), (No | 51520 | lstm_1[0][0] |
| lstm_3 (LSTM) | [(None, None, 80), ( | 121920 | embedding_1[0][0]<br>lstm_2[0][1]<br>lstm_2[0][2] |
| attention_layer (AttentionLayer | ((None, None, 80), ( | 12880 | lstm_2[0][0]<br>lstm_3[0][0] |
| concat_layer (Concatenate) | (None, None, 160) | 0 | lstm_3[0][0]<br>attention_layer[0][0] |
| time_distributed (TimeDistribut | (None, None, 37429) | 6026069 | concat_layer[0][0] |

Figure 4: LSTM model Architecture

- Latent dimension for LSTM is 80 and embedding dimension is 300
- Three layer of encoder LSTM is created and one decoding layer.
- Attention is applied on encoder and decoder outputs
- Decoder and attention outputs are concatenated and passed to time distributed layer and then dense layer.
- Optimizer used is rmsprop, loss used is sparse_categorical_crossentropy.
- Reduce learning rate method is used to reduce the learning rate if the learning rate is stagnant or if there are no major improvements in training or if plateau is encountered while training the model.
- Early stopping is used to stop the training if there are no further reduction in the validation loss
- Inference is used to get the predicted output of the model given a input review.

- **Layers of LSTM model**
  - **Bahdanau attention layers**
    - It is also known as Additive attention as it performs a linear combination of encoder states and the decoder states.
    - All hidden states of the encoder (forward and backward) and the decoder are used to generate the context vector, unlike how just the last encoder hidden state is used in seq2seq without attention.
    - The attention mechanism aligns the input and output sequences, with an alignment score parameterized by a feed forward network.
    - It helps to pay attention to the most relevant information in the source sequence.
    - The model predicts a target word based on the context vectors associated with the source position and the previously generated target words.
  - **Alignment score**
    - The alignment score maps how well the inputs around position "j" and the output at position "i" match. The score is based on the previous decoder's hidden state, s just before predicting the target word and the hidden state, h of the input sentence.
    - The formula for alignment score

$$e_{ij.} = a(s_{i-1}, h_j) \qquad \text{Alignment Score}$$

Figure 5: Alignment Score

- The decoder decides which part of the source sentence it needs to pay attention to, instead of having encoder encode all the information of the source sentence into a fixed length vector.
- The alignment vector that has the same length with the source sequence and is computed at every time step of the decoder.

– **Attention weights**

- We apply a softmax activation function to the alignment scores to obtain the attention weights.
- Attention weights formula

$$\alpha_{i_j} = {\exp(e_{ij})}/{\sum_{k-1}^{Tx} \exp(e_{ik})} \quad \text{Attention weight}$$

Figure 6: Attention weights

- Softmax activation function will get the probabilities whose sum will be equal to 1, This will help to represent the weight of influence for each of the input sequence.
- Higher the attention weight of the input sequence, the higher will be its influence on predicting the target word.

– **Context Vector**

- The context vector is used to compute the final output of the decoder. The context vector is the weighted sum of attention weights and the encoder hidden states (h, h, . . . ,h), which maps to the input sentence.
- Context vector formula

$$C_i = \sum_{j=1}^{Tx} \alpha_{ij}\, h_j \quad \text{Context vector}$$

Figure 7: Context vector

– **Predicting the target word**

- To predict the target word, the decoder uses Context vector, Decoder's output from the previous time step (y) and Previous decoder's hidden state (s) to get the output.
- Predict method formula

$$s_i = f(s_{i-1}, c_i, y_{i-1})$$

Figure 8: Predict formula

– **Bahdanau uses only concat score alignment model**

**Bahdanau** uses only concat score alignment model

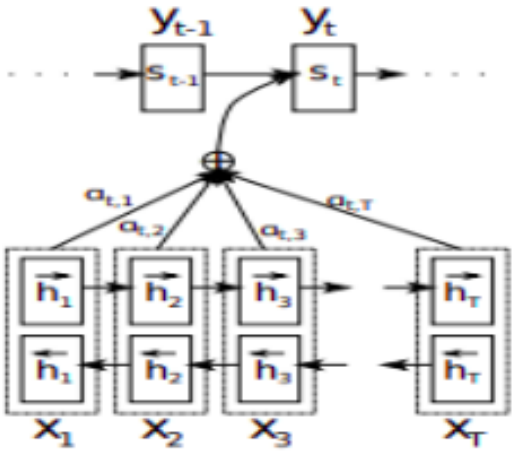


Figure 1: The graphical illustration of the proposed model trying to generate the $t$-th target word $y_t$ given a source sentence $(x_1, x_2, \ldots, x_T)$.

Figure 9: Bandanu model

- **Time distributed layer**
  - In Machine Learning or Deep learning, one can now predict values on complex data by using Neural Networks.
  - And for the majority of them, one will send one or several inputs to be analysed at a given time.
  - Sometimes, these values are chronological in order. For example stock prices in time, video frames, or human size at a certain age in its life.
  - For this kind of data, we already have some nice layers to treat data in the time range, for example, LSTM.
  - But what if you need to adapt each input before or after this layer? This is where Time Distributed layer can give a hand.
  - It synchronizes the inputs coming from the output of previous layer then a dense layer, this is the classification layer then a softmax activation function is applied to get the output.

- The decoding stage for the LSTM is the basic greedy deocder which uses the keras model.predict and argmax function to get the output until <EOS> token or stop condition is encountered.

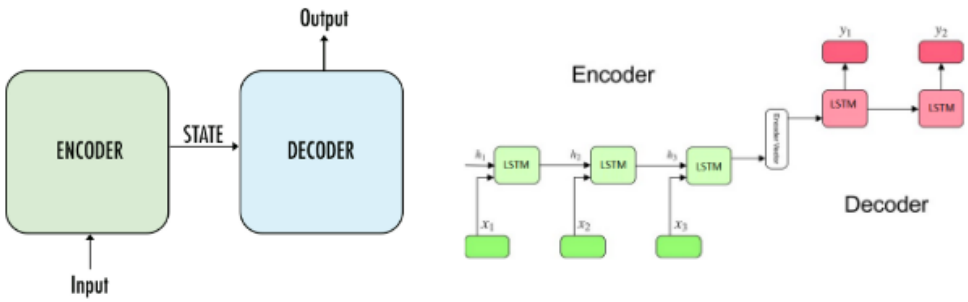


Figure 10: Encode Decoder Block Diagram

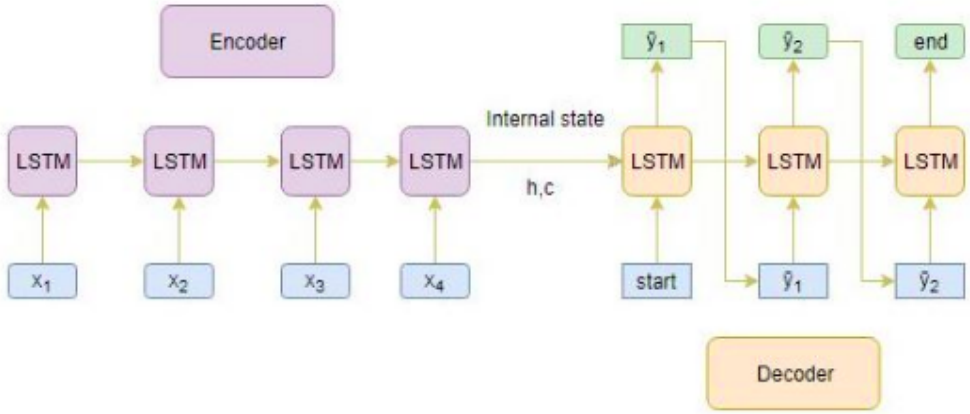


Figure 11: Inference Block Diagram

- **T5-small transformer model**
  - Transfer learning, where a model is first pre trained on a data rich task before being fine tuned on a downstream task, has emerged as a powerful technique in natural language processing (NLP).
  - T5 is an encoder decoder model pre trained on a multi task mixture of unsupervised and supervised tasks and for which each task is converted into a text to text format.
  - T5 works well on a variety of tasks out of the box by pre pending a different prefix to the input corresponding to each task, e.g., for translation: translate English to German: . . . , for summarization: summarize: . . . .
  - T5 uses relative scalar embedding.
  - T5 small model diagram

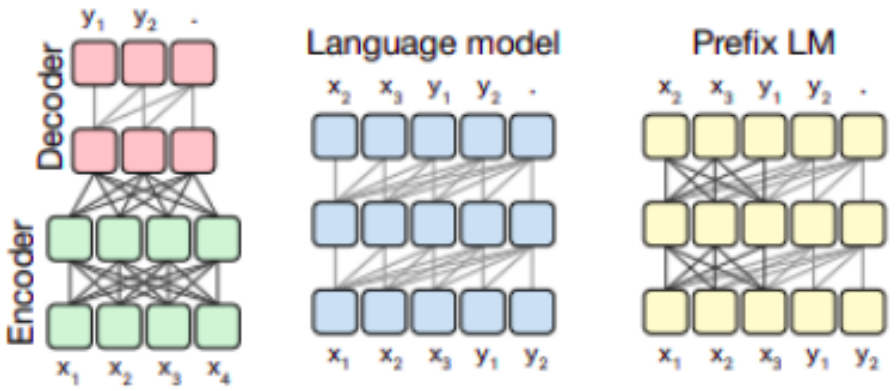


Figure 12: T5 small model

- T5 is an encoder decoder model and converts all NLP problems into a text to text format.
- It is trained using teacher forcing.
- This means that for training we always need an input sequence and a target sequence.
- The input sequence is fed to the model using input_ids.
- The target sequence is shifted to the right, i.e., pre pended by a start sequence token and fed to the decoder using the decoder_input_ids.
- In teacher forcing style, the target sequence is then appended by the EOS token and corresponds to the labels.
- The PAD token is hereby used as the start sequence token.
- T5 can be trained/fine tuned both in a supervised and unsupervised fashion.
- In this setup the input sequence and output sequence are a standard sequence to sequence input output mapping.
- The decoder stage is simply uses the inbuilt function given by T5small tokenizer for getting the output.

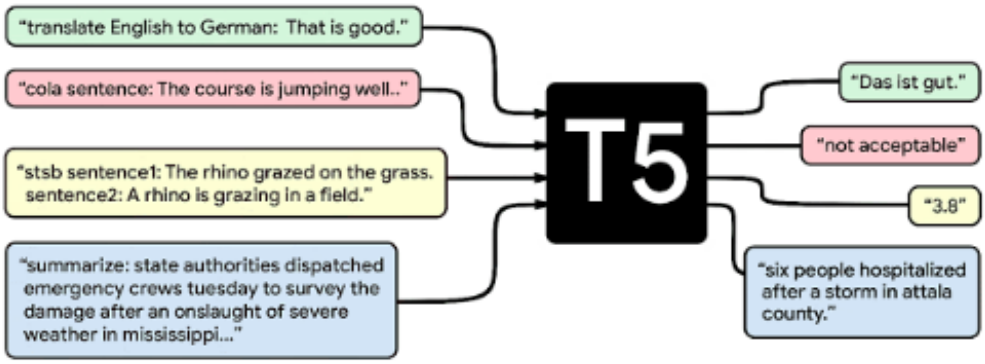


Figure 13: Prefix translate and summarize supported by T5small

# 4 Experiment

Two models were implemented one model from scratch using LSTM based seq2seq2 encoder decoder using attention and using pre trained T5 tokenizer the hyper parameters common in both models are maximum review length max_rl=80, maximum summary length max_sl=10, maximum UNK token in review unk_rl=1, maximum UNK token in summary unk_sl=0 and threshold=20. Additional parameters for LSTM model epochs=100, optimizer=Rmsprop, initial learning_rate=0.001 and batch_size=512. Additional parameters for T5-small batch_size=128, shuffle_size=1024, epochs=150, optimizer=Adam, initial learning_rate=1e-4.

## 4.1 Time consumed by LSTM and T5 model

- Total time required for training 64167.45829248428
- Total time to complete testing 15829.7639939785
- Total time required for completing whole process 80087.67300844193
- It can be seen that for more than 1,00,000 rows it take almost 1 and half day to complete the whole process
- T5 model takes at least 8 hours to complete whole process as it runs on GPU and as i used google colab it depends on availabilty of GPU is two or three GPU are available then whole process can be completed within 8 hours.
- Approximately this is ms/batch 248.44 the amount of time taken per batch.

## 4.2 Quantitative Analysis

For evaluation, metrics used are ROUGE-1, ROUGE-2 ROUGE-L and bleau score.

- Rouge is a standard metrics for summarization tasks; it measures similarity between reference summary and test summary.
- ROUGE-N is recall oriented measure; it divides the number of matching n-grams by the number of total n-grams in the reference summary.

$$ROUGE - N = \frac{\sum_{S \in ReferenceSummaries} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in ReferenceSummaries} \sum_{gram_n \in S} Count(gram_n)}$$

Figure 14: Rouge-N score

- ROUGE-L compares the longest common sequence (LCS) between reference summary and test summary. It eliminates the need to pre define n-gram length. The intuition is that the longer the LCS is, the more similar those two text are. Given Xreference summary X of length m and test summary Y of length n, it then computes an F-measure.

$$R_{lcs} = \frac{LCS(X,Y)}{m}$$
$$P_{lcs} = \frac{LCS(X,Y)}{n}$$
$$F_{lcs} = \frac{1 + \beta^2 R_{lcs} P_{lcs}}{R_{lcs} + \beta^2 P_{lcs}}$$

Figure 15: Rouge-L score

- Bleau score, or the Bilingual Evaluation Understudy, is a score for comparing a candidate translation of text to one or more reference translations. Although developed for translation,

it can be used to evaluate text generated for a suite of natural language processing tasks like text summarization and others.

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

Figure 16: Bleau score

### 4.3 Qualitative Analysis

Visual inspection the results

- **Below is the LSTM model output**

| | | |
|---|---|---|
| Review: hides spare tire wonderfully sweater dress bought clingy added support makes feel confident ride stays place feel overheated | Original: fantastic | Predicted: great |
| Review: returning thicker envisioning product good example 34 pay 34 embossing crooked cover squared corners | Original: uneven | Predicted: not bad |
| Review: smaller size indicates narrow wore work gave amish friend smaller feet likes | Original: smaller and more narrow than the sizing indicates | Predicted: too small |
| Review: purchases shoes family recent cruise perfect choice reviews correct run large men like run men sizes women size 8 1 2 order size 7 wear women 9 order 7 8 wear men 9 order 9 shoes extremely comfortable hikes kayaking recommend | Original: bargain | Predicted: great shoe |
| Review: fit bright white wanted want summer like fabric snag easily hangnail little burr shoe buckle instant run | Original: very white snags easily | Predicted: nice |

- **Below is the T5 model output**

| | | |
|---|---|---|
| Review: bought 4 yr old daughter dance class wore today time teacher thought adorable bo | Original: very cute | Predicted: the boots fit as expected but the boots are |
| Review: crocs general wife hard right size fit unisex shoes tailored men women purchase | Original: bought for my wife | Predicted: crocs are the best cro |
| Review: bought 5 red cute totally loves wants wear single day sensitive shoes clothes aw | Original: the best sandal ever | Predicted: i love these shoes i am |
| Review: picked jeans reviews advice buyers certainly correct tried jeans washing felt gr | Original: real jeans at a remarkable price | Predicted: the best selling jeans i have ever owned |
| Review: got today birthday wow cozy super shocked gift wanted uggs money reading plan pu | Original: best birthday gift ever | Predicted: gift for a man s birthday gift |

- **Loss and accuracy curves for LSTM model**

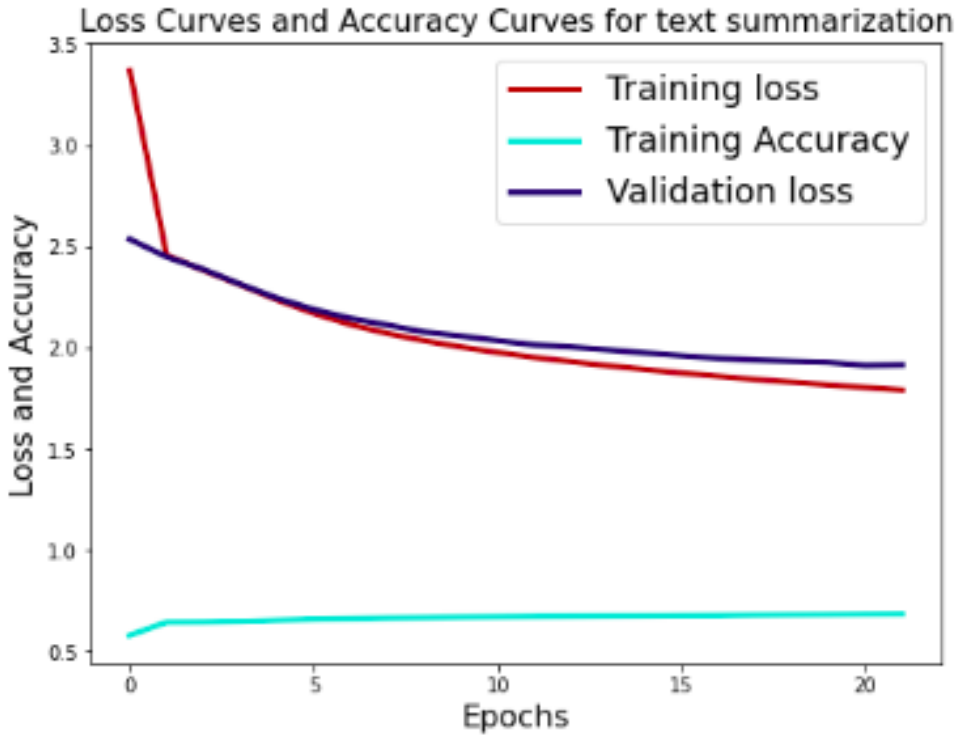


Figure 17: Loss and Accuracy curves during training for LSTM model for the amount of data used as per the snapshot displayed in figure 2

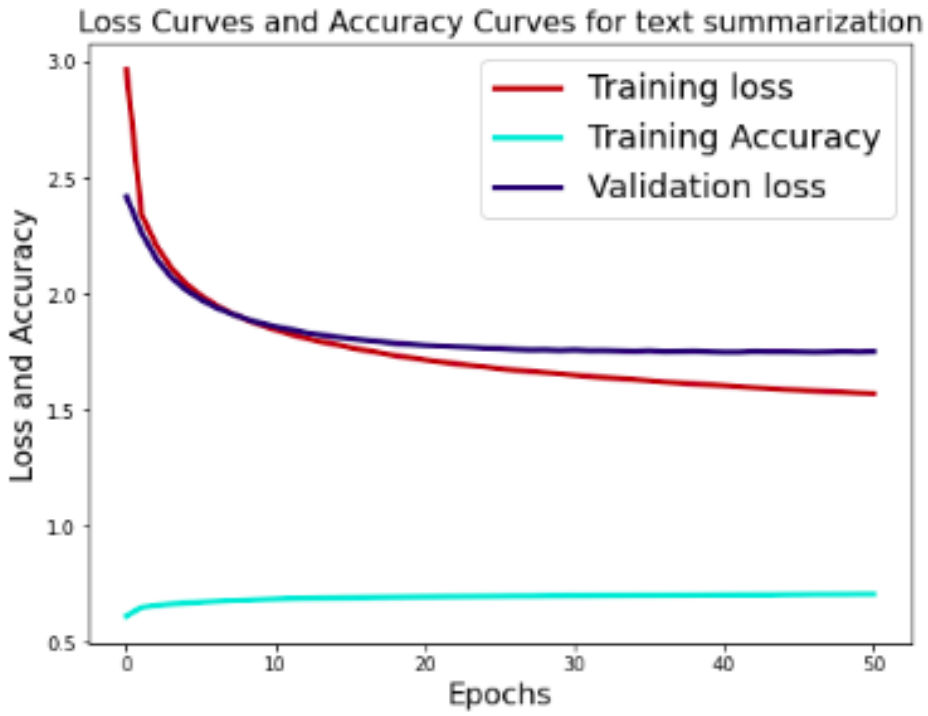


Figure 18: Loss and Accuracy curves during training for LSTM model for the amount of data used as per the snapshot displayed in figure 3

- **Output at epochs during training**

```
Epoch 1/100
223/223 [==============================] - 1248s 6s/step - loss: 4.1427 - accuracy: 0.5584 - val_loss: 2.4180 - val_accuracy: 0.6474
Epoch 2/100
223/223 [==============================] - 1244s 6s/step - loss: 2.3772 - accuracy: 0.6480 - val_loss: 2.2641 - val_accuracy: 0.6533
Epoch 3/100
223/223 [==============================] - 1243s 6s/step - loss: 2.2321 - accuracy: 0.6559 - val_loss: 2.1500 - val_accuracy: 0.6627
Epoch 4/100
223/223 [==============================] - 1239s 6s/step - loss: 2.1236 - accuracy: 0.6631 - val_loss: 2.0704 - val_accuracy: 0.6679
Epoch 5/100
Epoch 46/100
223/223 [==============================] - 1281s 6s/step - loss: 1.5802 - accuracy: 0.7056 - val_loss: 1.7491 - val_accuracy: 0.7001
Epoch 47/100
223/223 [==============================] - 1270s 6s/step - loss: 1.5774 - accuracy: 0.7062 - val_loss: 1.7482 - val_accuracy: 0.7000
Epoch 48/100
223/223 [==============================] - 1273s 6s/step - loss: 1.5757 - accuracy: 0.7063 - val_loss: 1.7489 - val_accuracy: 0.6999
Epoch 49/100
223/223 [==============================] - 1275s 6s/step - loss: 1.5671 - accuracy: 0.7072 - val_loss: 1.7504 - val_accuracy: 0.6997
Epoch 50/100
223/223 [==============================] - 1270s 6s/step - loss: 1.5673 - accuracy: 0.7069 - val_loss: 1.7490 - val_accuracy: 0.7004
Epoch 51/100
223/223 [==============================] - 1268s 6s/step - loss: 1.5697 - accuracy: 0.7066 - val_loss: 1.7504 - val_accuracy: 0.6997
Epoch 00051: early stopping
total time required for training  64167.45829248428
```

Figure 19: Output at epochs during training for LSTM

```
| epoch   0 | [  199/  809] | ms/batch 247.76 | loss  4.32 | val loss  4.37
| epoch   0 | [  399/  809] | ms/batch 248.13 | loss  3.74 | val loss  4.19
| epoch   0 | [  599/  809] | ms/batch 248.59 | loss  4.19 | val loss  4.01
| epoch   0 | [  799/  809] | ms/batch 247.97 | loss  3.83 | val loss  3.94
| epoch   1 | [  199/  809] | ms/batch 248.90 | loss  3.82 | val loss  3.87
| epoch   1 | [  399/  809] | ms/batch 249.66 | loss  3.50 | val loss  3.90
| epoch 146 | [  799/  809] | ms/batch 249.39 | loss  0.78 | val loss  5.40
| epoch 147 | [  199/  809] | ms/batch 249.74 | loss  0.71 | val loss  5.46
| epoch 147 | [  399/  809] | ms/batch 249.19 | loss  0.62 | val loss  5.51
| epoch 147 | [  599/  809] | ms/batch 249.66 | loss  0.84 | val loss  5.53
| epoch 147 | [  799/  809] | ms/batch 248.74 | loss  0.77 | val loss  5.41
| epoch 148 | [  199/  809] | ms/batch 251.35 | loss  0.73 | val loss  5.56
| epoch 148 | [  399/  809] | ms/batch 249.05 | loss  0.66 | val loss  5.45
| epoch 148 | [  599/  809] | ms/batch 249.06 | loss  0.69 | val loss  5.33
| epoch 148 | [  799/  809] | ms/batch 248.31 | loss  0.87 | val loss  5.32
| epoch 149 | [  199/  809] | ms/batch 249.31 | loss  0.71 | val loss  5.50
| epoch 149 | [  399/  809] | ms/batch 248.14 | loss  0.51 | val loss  5.57
| epoch 149 | [  599/  809] | ms/batch 250.74 | loss  0.73 | val loss  5.53
| epoch 149 | [  799/  809] | ms/batch 248.44 | loss  0.84 | val loss  5.47
```

Figure 20: Output at epochs during training for T5 small

- **Rouge AND Bleau Scores**

- textIf i use 117799 rows of data for training or 114079 rows of data for training the score remain same there is no significant difference between them, there are pdf file of runs and it can be seen in that too for further details

```
bleu, precisions, bp, ratio, translation_length, reference_length (0.0,
[0.2900551776136539, 0.0, 0.0, 0.0], 1.0, 19.09509591394331, 466856,
24449)
bleau score 0.7338717254431542
rouge2 (0.06599443828484215, 0.8312236286919831, 0.03436126421544687)
rouge {'rouge_1/f_score': 0.36915419958439477,
       'rouge_1/r_score': 0.36583216309687744,
       'rouge_1/p_score': 0.41662537566051006,
       'rouge_2/f_score': 0.2881554948360078,
       'rouge_2/r_score': 0.33457293997806903,
       'rouge_2/p_score': 0.2751092653511975,
       'rouge_l/f_score': 0.6180779054304887,
       'rouge_l/r_score': 0.6641052933641588,
       'rouge_l/p_score': 0.5875250521493721}
```

Figure 21: Scores for LSTM

bleu, precisions, bp, ratio, translation_length, reference_length (0.0, [0.22144055961974557, 0.0, 0.0, 0.0], 1.0, 35.20658547586829, 780530, 22170)

bleau score 0.6859844838250574

rouge2 (0.16848225200975267, 0.15806677562456223, 0.18036717690370083)

rouge {'rouge_1/f_score': 0.04740721897690091,
'rouge_1/r_score': 0.06681084477092596,
'rouge_1/p_score': 0.04179674019660488,
'rouge_2/f_score': 0.005587580152682946,
'rouge_2/r_score': 0.009073214912186496,
'rouge_2/p_score': 0.004661490216294005,
'rouge_l/f_score': 0.036532753369628,
'rouge_l/r_score': 0.06430128659492665,
'rouge_l/p_score': 0.03582168095023305}

Figure 22: Scores for T5 small

## 5 Discussion/conclusion

It can be observed from the score that LSTM based model gives the best output also it can be seen in qualitative analysis part . One of the reason for the LSTM model outperforming the T5-small with a small margin, it could be the use of pre trained conceptual numberbatch word embeddings, these embeddings have set bench marks for many NLP related tasks, also the attention mechanism used, some time its not required to apply attention on all the states just by applying attention to few states only it can provide human like output. Additionally the LSTM model converges faster than the T5-small model the loss remains same in T5-small to 2.74 or 3.20 and doesn't reduce further.

To further improve the output mutliheaded attention can be used with fixed number of window size to pay attention to specific states, pre trained model can be further fine tuned with pre trained embedding like numberbatch, temperature based decoding scheme can be used instead of greedy or beam search based, NER with context free grammar can be incorporated during attention stage with input sequence, also extractive and abstractive model can be combined and Custom Adaptive attention based on length of target sequence can be designed. very limited research is done on this area by combining different models and experimenting the performance.

## 6 future work

Multiple architecture or models need to be combined like pointer based methods, multihead attention, combining pre trained models with scratch model and fine tuning these are all different experiments that can be carried to measure the performance.

Also the evaluation metrics need to be tuned as they don't understand the synonyms, antonyms and entity so if the output is a synonym of predicted the metrics give score zero but in actual it should be one. So metrics like rouge bleau should be upgraded so one can trust the evaluations and it also can help in cases when the target variable is not defined that is unsupervised learning.

## References


[1] Luhn, H. P. (1958) The automatic creation of literature abstracts, IBM Journal of Research and Development, vol. 2, no. 2.

[2] Salton G. Buckley C. (1988) Term-weighting approaches in automatic text retrieval. Information Pro- cessing and Management, vol. 24, pp. 513 523.

[3] Lin, C. Hovy, E. (2000) The automated acquisition of topic signatures for text summarization. Proceed- ings of the International Conference on Computational Linguistic.

[4] Cho, K. et al. (2014) Learning Phrase Representations using RNN Encoder Decoder for Statistical Ma- chine Translation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).

[5] Bahdanau, D. et al. (2015) Neural Machine Translation by Jointly Learning to Align and Translate. Pro- ceedings of the 2015 Conference on International Conference on Learning Representation (ICLR).

[6] Chopra, S. Mozer, M.C. (2016) Abstractive Sentence Summarization with Attentive Neural Networks. Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology.

[7] Lin, C. (2004) ROUGE: A Package for Automatic Evaluation of Summaries.Proceedings of Workshop on Text Summarization Branches Out, Post-Conference Workshop of ACL 2004. Barcelona, Spain.

[8] Abstractive Text Summarization using Sequence to sequence RNNs and Beyond Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, Bing Xiang