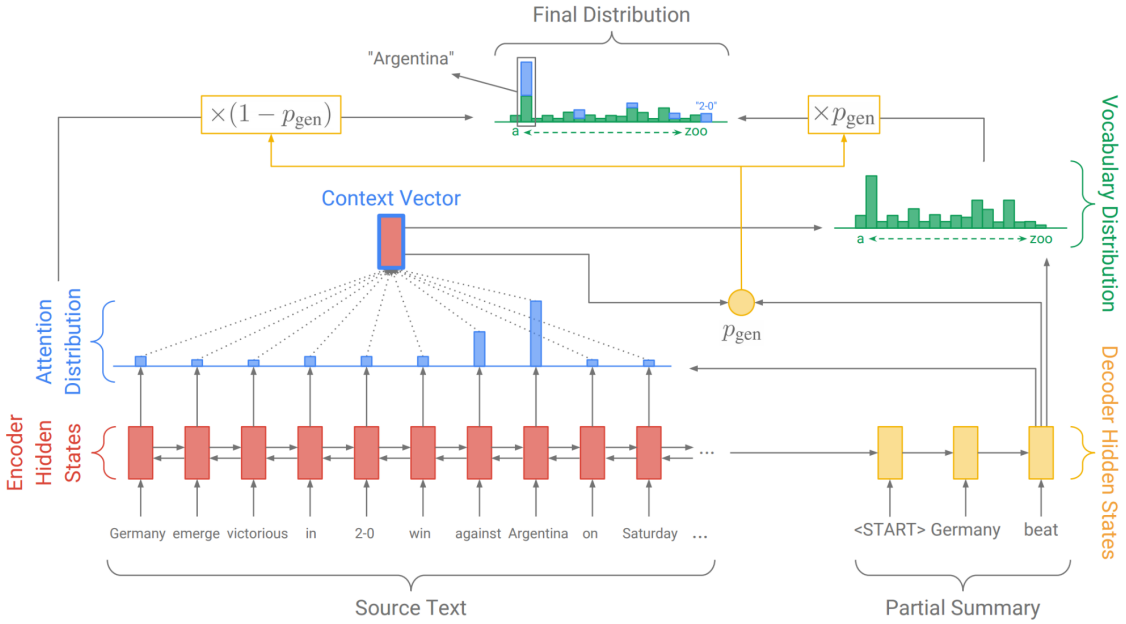
Abstractive Text Summarization Using Deep Learning

CIE 555 : Neural Networks and Deep Learning Course project



Mahmoud Yasser **201700960**

Ahmed Abdelhafez **201700448**

Ahmed Samy **201700259**

19.05.2021

4TH Year Communications and Information Engineering

University of Science and Technology

Zewail City

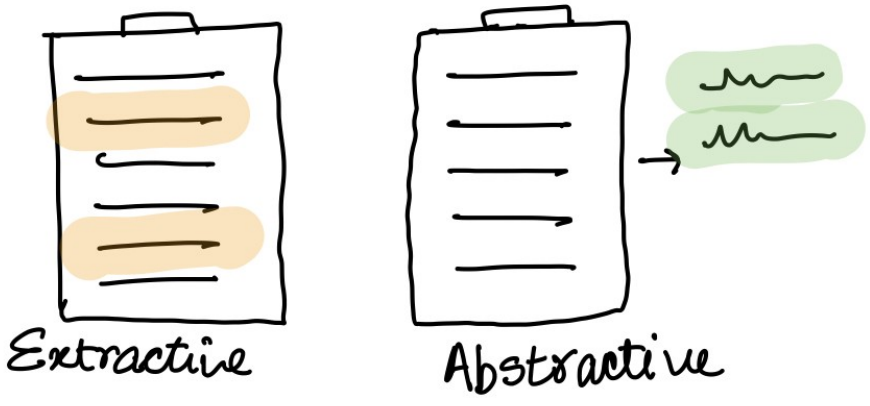
Problem definition and motivation

Summarization of a text is the process of producing a short version of the text that delivers the same information of the original text. Text summarization can be applied in wide range of use cases which include but not limited to:

- Automated content creation.
- Email summarization.
- Summarizing medical reports.
- Internal document workflow, summarizing unstructured data which helps analysts understand the documents more easily.
- Media monitoring, by summarizing the contents of posts and comments to the reviewers.

There are two types of text summarization: extractive and abstractive.

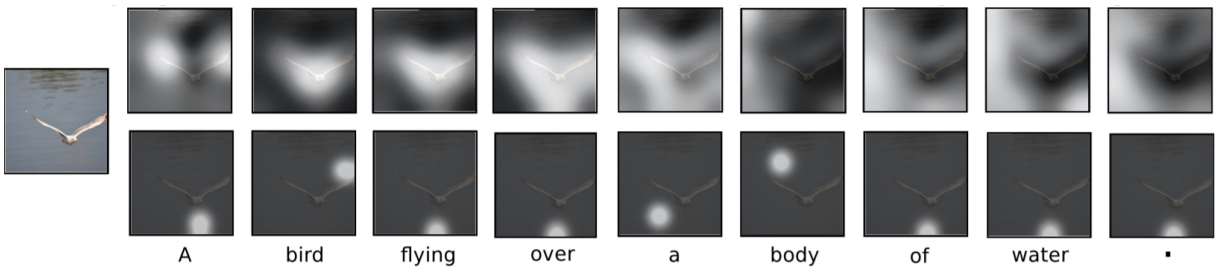
- Extractive summarization:
  This type of summarization extracts the most important sentences in the text and produces them as the summary. So, the extracted summary will necessarily be part of the original text. This is done by assigning a score to each sentence in the text and then outputs the sentences with a relatively high score. This type of summarization performs well using classical machine learning algorithms.
- Abstractive summarization:
  This type of summarization produces a new text that is not necessarily part of the original text. The main algorithm of this type of summarization is encoder-decoder structures in which Deep learning architectures are used like RNNs (LSTM, GRU or AMR) along with attention mechanisms.



Abstraction is harder than extraction for both humans and computers. So, the abstractive summarization is a much more difficult problem to handle than extractive summarization as it implies that the deep neural network has to learn the content of the text in order to summarize the meaning of it correctly with no grammatical or structural mistakes. So, in this project we would like to tackle the problem of abstractive summarization and its associated difficulties. This will help us extend our understanding to include more deep learning topics (seq-to-seq modeling with encoder-decoder architecture, attention mechanisms, coping mechanisms and Transformers).

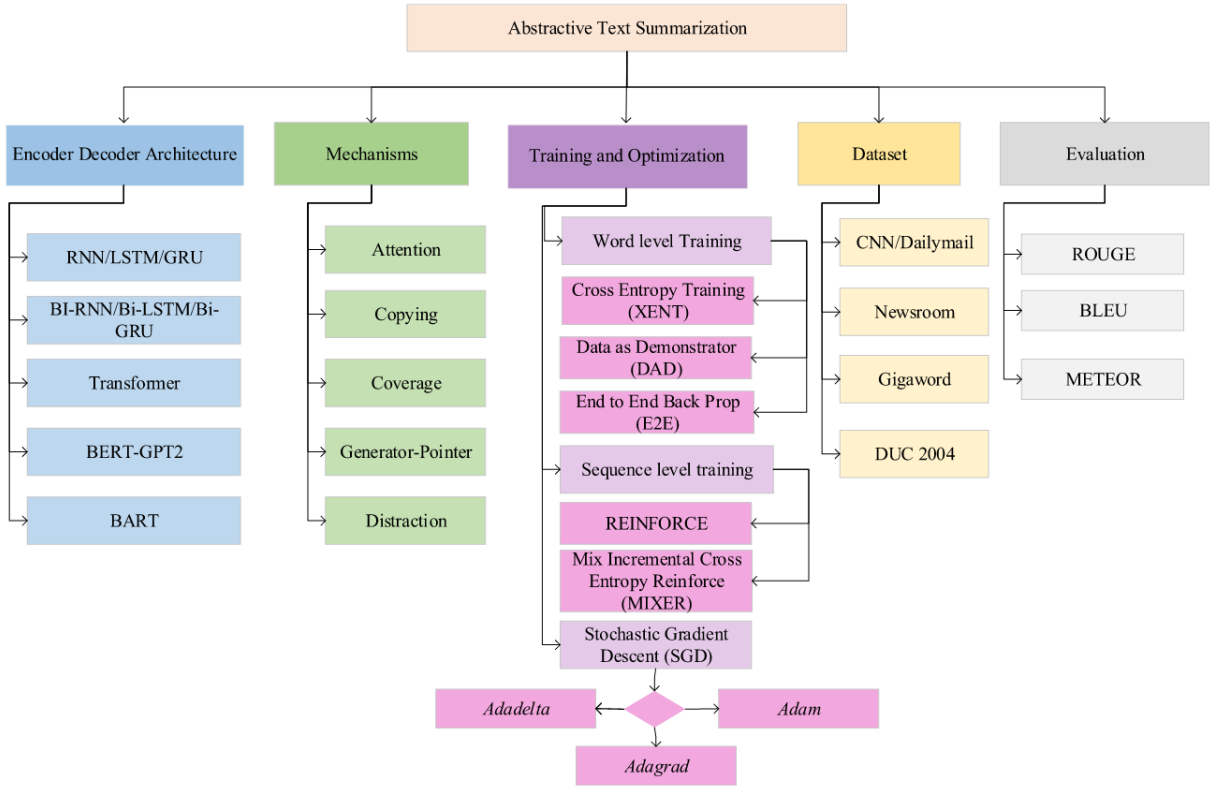
Literature Review

Before the introduction of encoder-decoder architecture in 2013, most of the abstractive summarization was with semantic methods used on ontology-based methods [1] and template-based methods [2]. The encoder-decoder architecture then came to be the actual beginning of neural networks approaches in machine translation that was the core of Google's Translate service [3] and extended later for text summarization. However, the encoder-decoder architecture was not able to capture the dependency across long texts, leading to improper summarization . As the internal state between the encoder and decoder was prone to change each new time step which loses the information about the far history that might be important to the summarization. This was a barrier until Bahdanau [4] introduced the attention mechanism in 2014, in which the decoder learns to focus its attention on certain parts of the encoded text that will produce the best decoded text. For example, in image capturing, the decoder will learn to focus its attention in specific parts of the image in order to produce the right caption.



The attention method that Bahdanau introduced was later known as additive attention as different types of attention mechanisms were introduced like the multiplicative attention by Luong [5] later in 2015, different types of attention mechanisms were also introduced for example, Hierarchical attention [6], Multi-headed self-attention [7] . After these two important papers in 2017, the complexity of encoder and decoder was simplified by only using attention (or self-attention) that does not use RNNs and only uses attention which is known now as Transformers [7]. Different architectures were later introduced like BERT-GPT2 and BART. The

work in the training method was very active too, the framework of abstractive text summarization is shown in the following figure[8].



The following table shows the results some selected papers in the past few years:

Encoder -- Decoder	Summary	Dataset	ROUGE Score
RNN -- LSTM	In 2016, S. Chopra [9] et al from Facebook used a simplified attention RNN encoder-decoder of Bahdanau's encoder-decoder for text summarization instead of Bahdanau's machine translation model. They outperform the state of the art back models back then.	DUC-2004, Gigaword corpus	28.97
Bi-RNN -- RNN	In 2016, J. Gu et al [10] proposed a new model called COPYNET with encoder-decoder structure. The new copying algorithm can choose sub-sequences in the input sequence and put them at proper places in the output sequence.	LCSTS	35.00
Transformers	In 2019, X. Duan et al [11] from China	Gigaword,	38.72

	proposed a contrastive attention mechanism for abstractive sentence summarization, using two attention layers: a conventional attention that does the regular job and a new opponent attention that penalizes the attention to irrelevant parts of the text.	Newsroom	
Bi-LSTM -- Bi-LSTM	In 2017, A. See et al [12] from Stanford Uni and Google constructed a hybrid pointer-generator architecture with coverage. Pointer-generator approach helped eliminate out of vocabulary (OOV) words. The coverage helps reduce repetition in the result text. Both are functionalities added to the basic neural encoder-decoder architecture just like the attention.	CNN/Daily mail	39.53
Bi-LSTM -- Bi-LSTM	In 2018, S. Gehrmann et al [13] from Harvard built a model that can identify phrases in the text that might be in the summary. This so-called content selector can be used for a bottom-up attention that restricts the ability of abstractive summarizers to copy words from the source.	CNN/Daily mail	41.22
Transformer	In 2020, I. Saito et al [14] introduced a model that has a word-level prototype extractor and a prototype-guided abstractive summarizer. This hybrid technique uses the prototype extractor to extract the important parts of the text then pass it to the abstractive summarizer.	CNN/Daily mail, LCSTS	44.79

Table(1).

In our project we intend to implement an architecture very similar to the architecture proposed in [9] in which a simplified version of the original Bahdanau's model is implemented. So we will implement an RNN based encoder and an LSTM based decoder with attention. We will also try extending the model with modern mechanisms like generator-pointer and coverage to see the effect of them on the summarized text.

Dataset

For the dataset, we will try the Amazon Fine Food Reviews [15] as it contains relatively small texts and their summary. Then we may use Gigaword when we are sure that our model works correctly.

Amazon Fine Food Reviews

This dataset consists of reviews of fine foods from amazon. The data was gathered in more than 10 years. It includes all ~500,000 reviews up to October 2012.
Data columns description :

Id	Row ID
ProductId	Unique identifier for the product
UserId	Unique identifier for the user
ProfileName	Profile name of the user
HelpfulnessNumerator	Number of users who found the review helpful
HelpfulnessDenominator	Number of users who indicated whether they found the review helpful or not
Score	Rating between 1 and 5
Time	Timestamp for the review
Summary	Brief summary of the review
Text	Text of the review

Table (2).

We will use the last two columns (Summary and Text) as our training and testing targets.

Objectives

1. Preprocessing of the data (cleaning, standardizing and embedding).
2. Implementing the base attention encoder-decoder architecture.
3. Hyper parameter Tuning (number of RNNs and LSTMs layers, Learning rate, optimizer and regularization)
4. If time allows we will try other recent mechanisms like Pointer-Generator and Corerage.
5. The baseline ROUGE-1, ROUGE-2, ROUGE-L is 28.97, 8.26, 24.06 respectively.

Snippet of the dataset:

	Summary	Text
0	Good Quality Dog Food	I have bought several of the Vitality canned d...
1	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...
2	"Delight" says it all	This is a confection that has been around a fe...
3	Cough Medicine	If you are looking for the secret ingredient i...
4	Great taffy	Great taffy at a great price. There was a wid...

Data Preprocessing

1. Dataset Cleaning
    a. First, we read the dataset and took only our parts of interest which are the Review text and the review summary.
    b. We removed duplicates and nulls.
2. Text preprocessing and Cleaning
    a. We start by making all text in lowercase and remove all non letter symbols like brackets ([]), dashes (-), apostrophe (‘s), and all punctuation and special characters.
    b. Remove stop words. Stop words are words that do not contribute to the meaning like tense-related and grammar-related words.
3. Determining our parameters of maximum text length and maximum summary length to be considered in our model design
    a. We analyze the length of the source and the target to get an overall idea about the distribution of length of the text.
    b. We found that 95% of Text data is below 100 words and 99.9% of the summary text is below 20 words. We took those as our parameters.

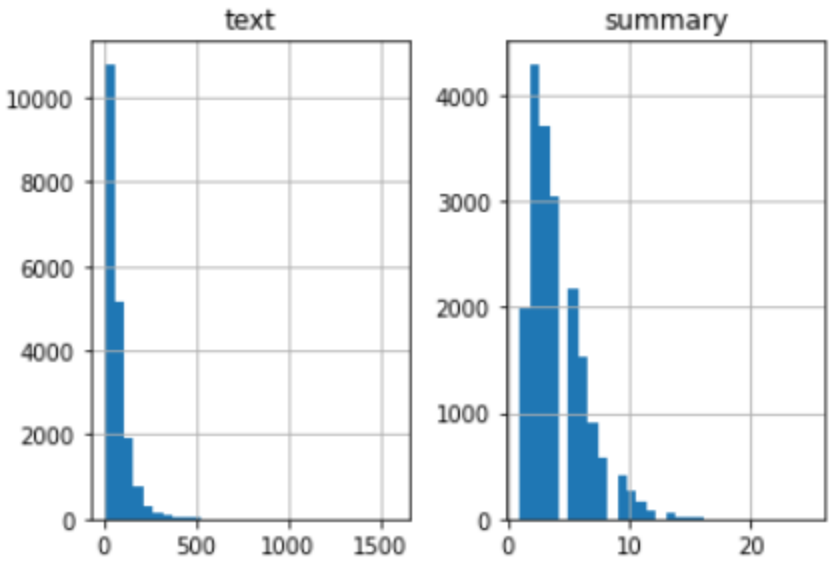


Figure (1).

c. Then, we dropped the samples of the data which are out of our parameter limits.

4. Making the data understandable for the model
    a. We start by tokenizing the data by fitting the tokenizer on the training set then tokenizing all test and training data.
    b. Finally, we padded all texts less than the maximum text length and the summaries less than the maximum summary size to meet the requirements and be fit to train the model.
    c. Number of vocab achieved in the texts is 16553
    d. Number of vocab achieved in the summaries is 5147

Models

LSTM encoder-decoder with attention layer:
we are building a 3 stacked LSTM for the encoder:

- Embedding Layer
- 3 LSTM

Attention Layer:

- 2 Dense layers : one for the encoder outputs, other for the Decoder outputs
- Additive attention

Decoder Layer:

- Embedding Layer
- 1 LSTM

A diagram of the full architecture is shown in figure [3]. The number of layers and parameters is shown in figure [2]

```
Model: "model_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_2 (InputLayer)            [(None, 100)]        0
__________________________________________________________________________________________________
embedding (Embedding)           (None, 100, 50)      827650      input_2[0][0]
__________________________________________________________________________________________________
lstm (LSTM)                     [(None, 100, 100), ( 60400       embedding[0][0]
__________________________________________________________________________________________________
input_3 (InputLayer)            [(None, None)]       0
__________________________________________________________________________________________________
lstm_1 (LSTM)                   [(None, 100, 100), ( 80400       lstm[0][0]
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, None, 50)     257350      input_3[0][0]
__________________________________________________________________________________________________
lstm_2 (LSTM)                   [(None, 100, 100), ( 80400       lstm_1[0][0]
__________________________________________________________________________________________________
lstm_3 (LSTM)                   [(None, None, 100),  60400       embedding_1[0][0]
                                                                 lstm_2[0][1]
                                                                 lstm_2[0][2]
__________________________________________________________________________________________________
bahdanau_attention (BahdanauAtt ((None, None, 100),  20100       lstm_3[0][0]
                                                                 lstm_2[0][0]
__________________________________________________________________________________________________
concat_layer (Concatenate)      (None, None, 200)    0           lstm_3[0][0]
                                                                 bahdanau_attention[0][0]
__________________________________________________________________________________________________
time_distributed (TimeDistribut (None, None, 5147)   1034547     concat_layer[0][0]
==================================================================================================
Total params: 2,421,247
Trainable params: 2,421,247
Non-trainable params: 0
__________________________________________________________________________________________________
```

Figure (2).

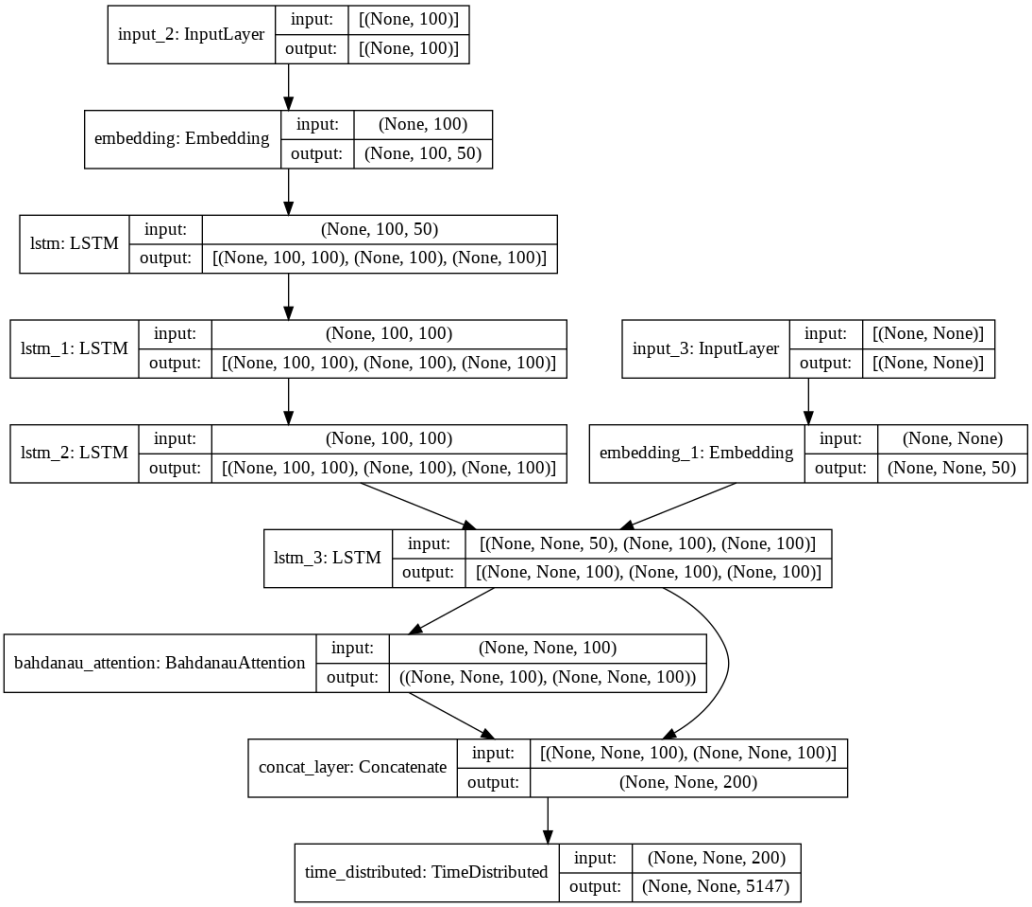


Figure (3).

GRU encoder-decoder with attention layer:
we are building a 3 stacked GRU for the encoder:

- Embedding Layer
- 3 GRU

Attention Layer:

- 2 Dense layers : one for the encoder outputs, other for the Decoder outputs
- Additive attention

Decoder Layer:

- Embedding Layer
- 1 GRU

A diagram of the full architecture is shown in figure [5]. The number of layers and parameters is shown in figure [4]

```
Model: "model_1"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
input_18 (InputLayer)        [(None, 200)]             0
_________________________________________________________________________________________________
embedding (Embedding)        (None, 200, 50)           1706900     input_18[0][0]
_________________________________________________________________________________________________
gru (GRU)                    [(None, 200, 200), (      151200      embedding[0][0]
_________________________________________________________________________________________________
input_19 (InputLayer)        [(None, None)]            0
_________________________________________________________________________________________________
gru_1 (GRU)                  [(None, 200, 200), (      241200      gru[0][0]
_________________________________________________________________________________________________
embedding_1 (Embedding)      (None, None, 50)          519200      input_19[0][0]
_________________________________________________________________________________________________
gru_2 (GRU)                  [(None, 200, 200), (      241200      gru_1[0][0]
_________________________________________________________________________________________________
gru_3 (GRU)                  [(None, None, 200),       151200      embedding_1[0][0]
                                                                   gru_2[0][1]
_________________________________________________________________________________________________
bahdanau_attention (BahdanauAtt ((None, None, 200),    80200       gru_3[0][0]
                                                                   gru_2[0][0]
_________________________________________________________________________________________________
concat_layer (Concatenate)   (None, None, 400)         0           gru_3[0][0]
                                                                   bahdanau_attention[0][0]
_________________________________________________________________________________________________
time_distributed (TimeDistribut (None, None, 10384)    4163984     concat_layer[0][0]
=================================================================================================
Total params: 7,255,084
Trainable params: 7,255,084
Non-trainable params: 0
```

Figure (4).

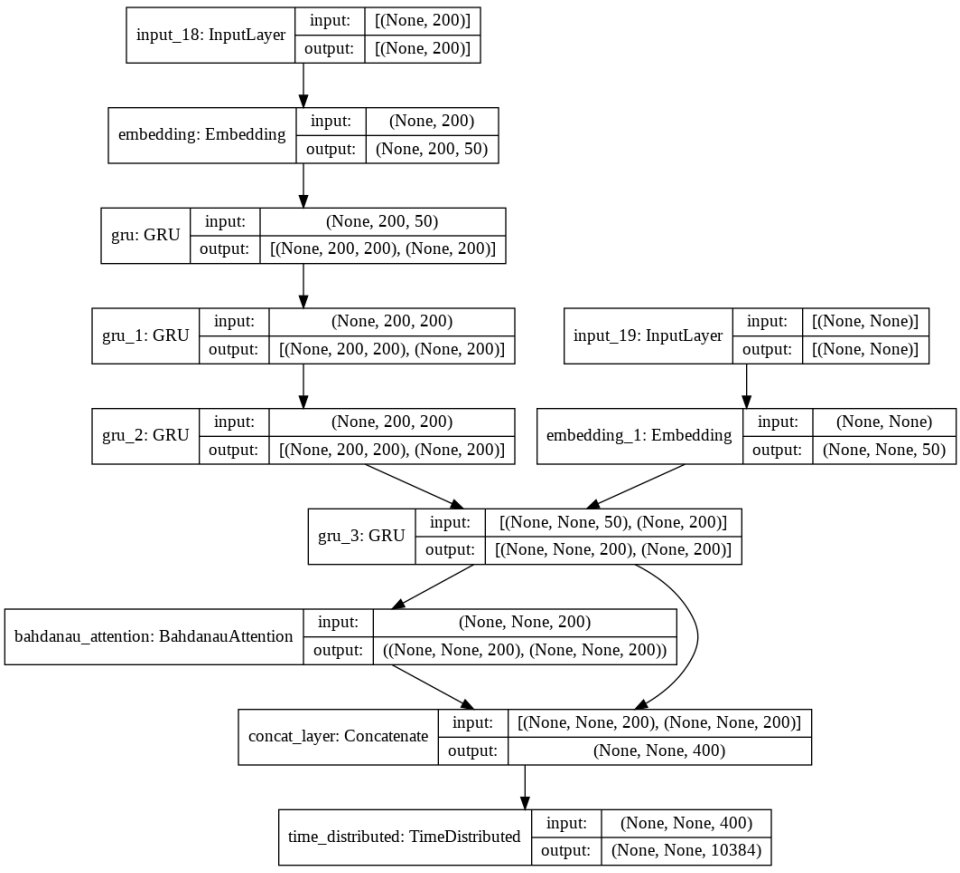


Figure (5).

RoBERTa Architecture:

RoBERTa "A Robustly Optimized BERT Pre-training Approach" is utilized in order to enhance our text summarization task. It is reported to outperform conventional LSTM and transformers [16]. Roberta is an optimized version of BERT "Bidirectional Encoder Representations from Transformers", which in contrast to sequence models, reads the input from both directions to boost the capacity of contextual learning. BERT-base [17] is an encoder architecture composed of 12 encoding layers, 768-hidden layers, 12 attention heads with 110 million parameters. It is pre-trained over BookCorpus(16 GP). The BERT encoder employs Masked Language Modelling (NLM) in which random words of the sequence are masked and the model is forced to predict the masked words. This enforces the model to learn contextual information from neighboring words. A binary classification loss, named as, Next Classification Prediction (NCP) is employed to enable BERT to model language across sentences. Different levels of embeddings are also used to embed word tokens (token embedding), position of words in the sequence which yields different tokenization for same words with different position since they can contribute differently to context, and segment embedding, specifying which segment this sentence is from. Figure 6 shows the multiple levels of embeddings in BERT. Figure 7 shows a high-level view of the BERT encoder utilizing Masked Language Modelling (MLM) and Next Classification Prediction (NCP).

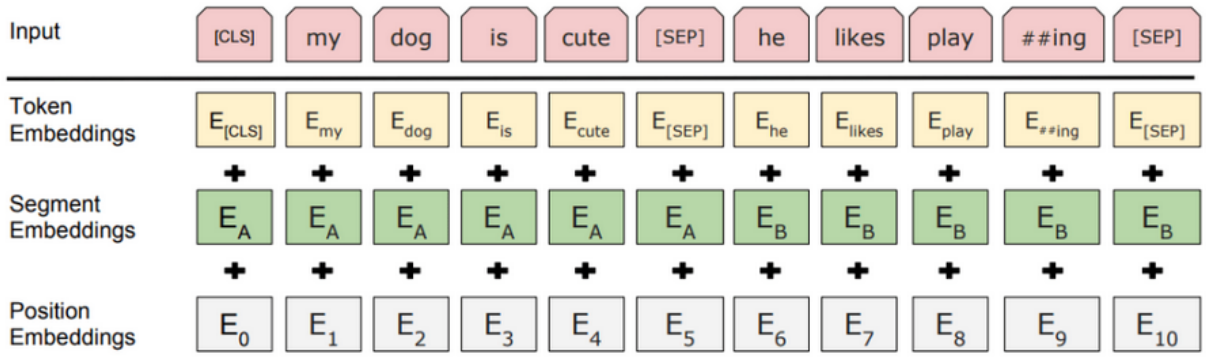


Figure (6)

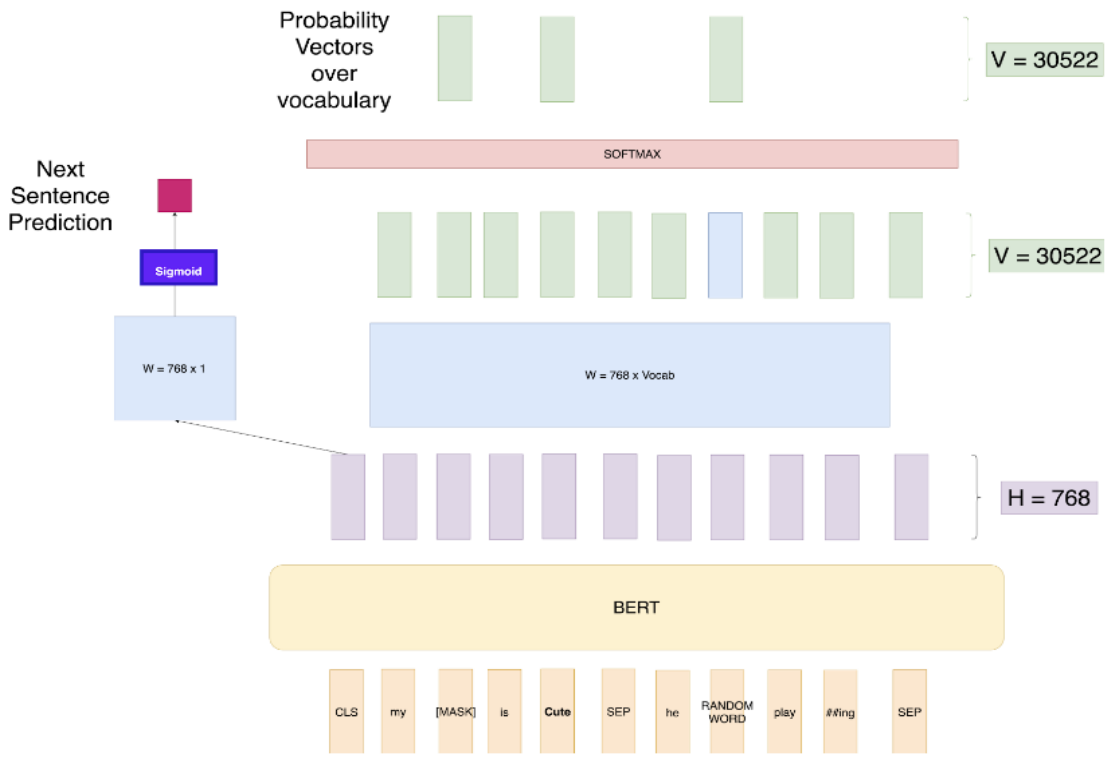


Figure (7)

In our work, we utilize Roberta, which is an optimized version of BERT. RoBERTa is pretrained on more training data (160 G versus 16 GB for BERT) with longer sequences, removes the Next Classification Prediction (NCP) loss with, and uses dynamic masking pattern instead of BERT's static masking pattern [16]. We fine tune two RoBERTa-to-RoBerta models to the text classification task for the Amazon Fine Food Reviews dataset (short sequences) and the CNN/Daily mail dataset (longer sequences). Our models are fine tuned using NVIDIA-SMI GPU, and 26 GB RAM.

Results

LSTM encoder-decoder with attention:

ROUGE	Precision	Recall	F-measure

ROUGE 1 SCORE	**18**	**13**	**14**
ROUGE 2 SCORE	**04**	**02**	**03**
ROUGE L SCORE	**18**	**13**	**14**

Table (3).

Sample 1:

Original Text: yes had the original turkish taffy no this is nothing like it the people who are writing the positive reviews are probably for the company that bought the brand name this is made with artificial flavor mono and di and egg albumin it soft and tastes fake any wax wrapped taffy that you buy in the store will probably be much better quality than this don't waste your time

Original Summary: nothing like the original

Predicted Summary: not so good

Sample 2:

Original Text: this is very good value use it to put in many asian recipes instead of chicken or vegetable broth the result is more authentic asian taste to my recipes

Original Summary: good product

Predicted Summary: great taste

RoBERTa:

ROUGE	Precision	Recall	F-measure
ROUGE 1 SCORE	**26.7**	**14**	**23.9**
ROUGE 2 SCORE	**11**	**10.2**	**10.1**
ROUGE L SCORE	**26.3**	**23.5**	**23.6**

Table (4).
Sample 1:

Original Text: I always have one or two bags of these treats in my cupboard, they are perfect for training. If you have a small dog you can cut them into smaller pieces so you can "jackpot" the

dog. My dog especially loves the peanut butter flavor but I buy all three. I highly recommend these!

Original Summary: THE BEST

Predicted Summary: Perfect training treats

Sample 2:

Original Summary: Mullah Omar, the reclusive founder of the Afghan Taliban, is still in charge, a new biography claims . An ex-Taliban insider says there have been rumors that the one-eyed militant is dead .

Predicted Summary: Miban Mohammed Omar is "still the leader" of the Islamic Emirate of Afghanistan. He is credited with founding the group in the early 1990s. The Taliban's "Cultural Commission" released the 11-page biography in several different translations. It appears to be aimed at dispelling rumors of Omar's death.

Hyperparameters Tuning

LSTM:

We used a Hyperband tuner with keras to tune the learning rate. The Hyperband tuner is a random search tuner but with early stopping, so it can get the results faster and avoid training models that are not promising

- First trial:

```
Trial 51 Complete [00h 14m 54s]
val_accuracy: 0.816554456949234

Best val_accuracy So Far: 0.8291195034980774
Total elapsed time: 01h 50m 11s

Search: Running Trial #52

Hyperparameter    |Value             |Best Value So Far
learning_rate     |0.00021375        |0.0085209
amsgrad           |True              |False
tuner/epochs      |40                |2
tuner/initial_e...|14                |0
tuner/bracket     |3                 |3
tuner/round       |3                 |0
tuner/trial_id    |08555bb256d4682...|None
```

We got learning rate of 0.00852
When we train the model it started to overfit:

```
Epoch 00006: saving model to /content/drive/MyDrive/NLP model
Epoch 7/50
2634/2634 [==============================] - 740s 281ms/step - loss: 0.8622 - val_loss: 0.9675

Epoch 00007: saving model to /content/drive/MyDrive/NLP model
Epoch 8/50
2634/2634 [==============================] - 741s 281ms/step - loss: 0.8482 - val_loss: 0.9709

Epoch 00008: saving model to /content/drive/MyDrive/NLP model
Epoch 9/50
2634/2634 [==============================] - 740s 281ms/step - loss: 0.8370 - val_loss: 0.9712

Epoch 00009: saving model to /content/drive/MyDrive/NLP model
Epoch 10/50
2634/2634 [==============================] - 740s 281ms/step - loss: 0.8288 - val_loss: 0.9773

Epoch 00010: saving model to /content/drive/MyDrive/NLP model
Epoch 11/50
2634/2634 [==============================] - 739s 281ms/step - loss: 0.8221 - val_loss: 0.9772

Epoch 00011: saving model to /content/drive/MyDrive/NLP model
Epoch 12/50
2634/2634 [==============================] - 740s 281ms/step - loss: 0.8137 - val_loss: 0.9825
```

- Second trial: In this part we used a 2 LSTM encoder and performed hyperparameters tuning on the dropout and the learning rate:

```
Trial 14 Complete [00h 02m 49s]
val_accuracy: 0.8069999814033508

Best val_accuracy So Far: 0.816789448261261
Total elapsed time: 00h 38m 39s

Search: Running Trial #15

Hyperparameter    |Value             |Best Value So Far
dropout_3         |0.25              |0.1
learning_rate     |0.0011972         |0.00029655
tuner/epochs      |2                 |2
tuner/initial_e...|0                 |0
tuner/bracket     |3                 |3
tuner/round       |0                 |0
```

We found that 0.1 dropout and 0.00296 learning rate were the optimal parameters When we trained the model it was converging slower (due to the low learning rate) but the variance was low:

```
Epoch 00001: saving model to /content/drive/MyDrive/NLP model
Epoch 2/50
2634/2634 [==============================] - 672s 255ms/step - loss: 1.2853 - val_loss: 1.2155

Epoch 00002: saving model to /content/drive/MyDrive/NLP model
Epoch 3/50
2634/2634 [==============================] - 667s 253ms/step - loss: 1.2036 - val_loss: 1.1521

Epoch 00003: saving model to /content/drive/MyDrive/NLP model
Epoch 4/50
2634/2634 [==============================] - 666s 253ms/step - loss: 1.1443 - val_loss: 1.1068

Epoch 00004: saving model to /content/drive/MyDrive/NLP model
Epoch 5/50
2634/2634 [==============================] - 666s 253ms/step - loss: 1.0943 - val_loss: 1.0735

Epoch 00005: saving model to /content/drive/MyDrive/NLP model
Epoch 6/50
2634/2634 [==============================] - 666s 253ms/step - loss: 1.0537 - val_loss: 1.0485

Epoch 00006: saving model to /content/drive/MyDrive/NLP model
Epoch 7/50
2634/2634 [==============================] - 666s 253ms/step - loss: 1.0208 - val_loss: 1.0312

Epoch 00007: saving model to /content/drive/MyDrive/NLP model
Epoch 8/50
2634/2634 [==============================] - 663s 252ms/step - loss: 0.9947 - val_loss: 1.0219

Epoch 00008: saving model to /content/drive/MyDrive/NLP model
Epoch 9/50
2634/2634 [==============================] - 660s 251ms/step - loss: 0.9748 - val_loss: 1.0074
```

The learning rate was tuned between 1e-4 and 1e-4 using log scale steps and the dropout was from 0 to 0.5 with a 0.05 step

Comments on results

We promised a performance that is at least on ROUGE-1, ROUGE-2, ROUGE-L scales are 28.97, 8.26, 24.06 respectively. However, for the LSTM based encoder-decoder with attention, we got at best 13, 4, 13 for ROUGE-1, ROUGE-2, ROUGE-L respectively. We think that due to:

1. Lack of resources:
    a. Lack of computational power: we were only using our GPU and google Colab notebooks. So, the time needed to train such models on such machinery would be very long and not compatible with the specified duration of the course.
    b. Lack of space: to achieve high performance with minimum validation loss, a quite enough number of examples should be provided in the training process. The training resources we have were getting out of memory when training to process the whole data. Also, processing more data will increase the training time greatly.

2. Unbalance of the dataset: we believe that the Amazon Fine Foods reviews data were slightly biased as it contains more frequent words than others (for example: the word “great” is appearing a lot). So, when the word “delightful” is in the target summary, the model most probability will out “great” not “delightful”, and since the ROUGE scores are mainly based on the n-grams recall, the recall of the target summary “delightful” will be zero on the . We believe that this causes the ROUGE score to be very low due to that fact. This can be improved by choosing a rich unbiased dataset (but that will come with its time and space cost).

References

[1] L. Chang-Shing, J. Zhi-Wei, and H. Lin-Kai, “A fuzzy ontology and its application to news summarization,” IEEE Trans. Syst. Man, Cybern.Part B, vol. 35, no. 5, pp. 859–880, 2005

[2] P.-E. Genest and G. Lapalme, “Framework for Abstractive Summarization using Text-to-Text Generation,” Work. Monolingual Text-To-Text Gener., no. June, pp. 64–73, 2011.

[3] Retrieved on 5/18 at 5 AM +2 GMT from: https://machinelearningmastery.com/encoder-decoder-recurrent-neural-network-models-neural-machine-translation/

[4] Bahdanau, D., Cho, K. H., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. Paper presented at 3rd International Conference on Learning Representations, ICLR 2015, San Diego, United States.

[5] Luong, M.-T., Pham, H. & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025

[6] R. Nallapati, B. Zhou, C. dos Santos, C. Gulcehre, and B. Xiang,‘‘Abstractive text summarization using sequence-to-sequence RNNs andbeyond,’’ inProc. 20th SIGNLL Conf. Comput. Natural Lang. Learn.,2016, pp. 280–290, doi:10.18653/v1/k16-1028

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez,L. Kaiser, and I. Polosukhin, ‘‘Attention is all you need,’’ inProc. Adv. Neural Inf. Process. Syst., NIPS, 2017, pp. 5998–6008.[Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[8] A. A. Syed, F. L. Gaol and T. Matsuo, "A Survey of the State-of-the-Art Models in Neural Abstractive Text Summarization," in IEEE Access, vol. 9, pp. 13248-13265, 2021, doi: 10.1109/ACCESS.2021.3052783.

[9] S. Chopra, M. Auli, and A. M. Rush, ‘‘Abstractive sentence summarization with attentive recurrent neural networks,’’ inProc. Conf. NorthAmer. Chapter Assoc. Comput. Linguistics: Human Lang. Technol., 2016,pp. 93–98.

[10] J. Gu, Z. Lu, H. Li, and V. O. K. Li, ‘‘Incorporating copying mechanism in sequence-to-sequence learning,’’ inProc. 54th Annu. Meeting Assoc.Comput. Linguistics (Long Papers), vol. 1, 2016, pp. 1631–1640, doi:10.18653/v1/p16-1154.

[11] X. Duan, H. Yu, M. Yin, M. Zhang, W. Luo, and Y. Zhang, ‘‘Contrastive Attention mechanism for abstractive sentence summarization,’’ inProc.Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf.Natural Lang. Process. (EMNLP-IJCNLP), 2019, pp. 3044–3053, doi:10.18653/v1/d19-1301.

[12] A. See, P. J. Liu, and C. D. Manning, ‘‘Get to the point: Summarization With pointer-generator networks,’’ inProc. 55th Annu. Meeting Assoc.Comput. Linguistics (Long Papers), vol. 1, 2017, pp. 1073–1083, doi:10.18653/v1/P17-1099.

[13] S. Gehrmann, Y. Deng, and A. Rush, ‘‘Bottom-up abstractive summariza-tion,’’ inProc. Conf. Empirical Methods Natural Lang. Process., 2018,pp. 4098–4109, doi:10.18653/v1/D18-1443

[14] I. Saito, K. Nishida, K. Nishida, A. Otsuka, H. Asano, J. Tomita, H. Shindo,and Y. Matsumoto, ‘‘Length-controllable abstractive summarization by guiding with summary prototype,’’ 2020,arXiv:2001.07331. [Online].Available: http://arxiv.org/abs/2001.07331

[15] McAuley, J. J., & Leskovec, J. (2013). From amateurs to connoisseurs. Proceedings of the 22nd International Conference on World Wide Web - WWW '13. https://doi.org/10.1145/2488388.2488466

[16] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov, “ RoBERTa: A Robustly Optimized BERT Pretraining Approach” arXiv:1907.11692v1

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding” arXiv:1810.04805v2