Abstractive Test Summarization Using Seq2Seq Deep Learning Models: Evaluation of Transfer Learning, Quantitative and Qualitative Metrics

Arjun Mahadevan, Anchal Jain, Nirosha Bugatha,
amahadev@sfu.ca, anchalj@sfu.ca, nbugatha@sfu.ca,

Ogheneovo Ojameruaye, Shahram Akbarinasaji
oojameru@sfu.ca, sakbarin@sfu.ca

Abstract

The project focuses on the application of sequence to sequence NLP models for abstractive summarization, where limited training data is available. We investigated two models and the variations observed in ROUGE metrics when applied to different datasets . We further showed examples of how it relates to qualitative summaries. The models were originally trained and tested to solve tasks such as news article summarization. We trained and tested the models on a text reviews dataset, followed by performance comparison of the two models and discussion on limitations in quantitative and qualitative metrics for summarization

1 Introduction

Document based text summarization has achieved significant interest in industry for text summarization tasks. The hope is to use state of the art models through transfer learning, that generalize well for different applications ranging from domain texts, reviews, news summaries to cases where limited training data is available. Abstractive text summarization aims to automatically generate a summary from a larger text or document while retaining its original information content. Summaries can have sentences that do not match the original text. Recent advances and progress in sequence to sequence models for language generation tasks, uses deep learning with different types of encoder-decoder and attention mechanisms that are discussed by Ruder[7], Dong et al.[1].

Once we have the abstract summaries, to assess the summary's content a variety of approaches have been adopted. Over the last decade, ROUGE and BLEU have become one of the most widely used standard automatic evaluation measure for evaluating summarization tasks[3][5]. In our study, we choose to use the ROUGE metric for discussing and comparing our experiments. ROUGE, or Recall-Oriented Understudy for Gisting Evaluation is a method to automatically determine the quality of a generated summary by comparing it to another set of gold standard summaries often created by humans. The measure is computed by counting the number of overlapping words or n-grams between the system-generated summary to be evaluated and the ideal summaries [3]. Various studies show that while ROUGE works well in capturing the n-gram overlap between human generated and automated summaries, there are still limitations in capturing the quality of the generated summaries from different aspects such as semantic quality and factual accuracy[4].

Language models usually require extensive training data and GPU resources to train from scratch for different applications. Current advances to use transfer learning through pre-trained Language Models like BERT, help alleviate some of the training time for several NLP tasks including language understanding, next word prediction, sentence prediction, question-answering and abstractive text summarization.

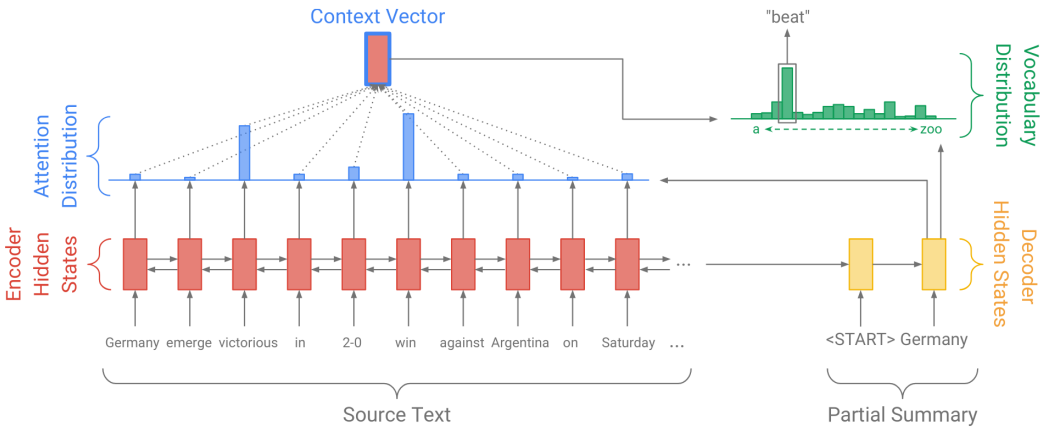

Figure 1: Basic framework for seq to seq models with attention[8]

In this project, we investigate transfer learning for abstract text summarization on different data sets, while using two state of the art models that are initialized using pre-trained language models and then fine-tuned. We experimented with two models(UNILM and Stacked Bidirectional LSTM), using three different data sets(CNN/Dailymail, GigaWords10K,and Amazon Reviews) to understand what transfer learning hyper parameters could be fine-tuned while preserving the models' architecture. The Amazon Reviews dataset is a new dataset that the authors have not used. We completed fine-tune training and testing on the two models using this dataset and address the limitations for transfer learning applications.

2 Model Architecture & Dataset

A general framework for sequence to sequence models is shown in Figure 1. The steps include cleaning the input text, segmentation, position embedding, and tokenization. Segmentation creates a pair of sentence to pass through the bidirectional LSTM transformer. Position embedding encodes each word sequence in the sentence, and tokenization creates tokens for each word in the sentence. A vector representation for each input is built using the segment embedding, position embedding and tokenized embedding. The bidirectional LSTM encoder network takes the entire input sequence one word at a time and is trained to capture the context. Pretrained word embeddings like the Bert Large, Wikipedia Corpus and vocabulary lists are used in the encoder for this purpose. The output from the encoder with start[SOS] and end tokens[EOS] are passed to the Decoder. The Decoder with attention layer is trained to predict a sequence of words with the highest probability. A set of probable sentences are built using the words and the beam size controls the number of probable sentence combinations. The final output sequence is the sentence with the word combination that has the highest probability. We picked two recent models which can be used for transfer learning that we hope will generalize well for different summarization tasks.

2.1 Model-A

The first model we used was the UNIfied pre-trained Language Model (UNILM) which can be adapted for different natural language processing tasks, released and published in October 2019[1]. The model architecture of UNILM follows that of BERT Large, uses English Wikipedia and Book Corpus. The details of the model transformers, layers and attention heads are presented by Li et al[1]. We maintained the same model architecture for our experimentation while fine-tuning: 1. The maximum sequence length (The maximum total input sequence length after WordPiece tokenization). Sequences longer than this will be truncated, and sequences shorter than this will be padded, 2. Maximum output target length, and 3. Beam Size for inference. The model was built and setup to run on Pytorch.

2.2 Model-B

The second model is based on a similar architecture using stacked bidirectional LSTM encoding, unidirectional LSTM decoding with Bahdanau Attention, and beam search that was adapted from the git hub repository[2]. The model used Glove pre-trained vectors for word embeddings and was setup to run on Tensorflow. For comparison to methods used in Model 1, we fine tuned two of the parameters for our transfer learning application: 1. Maximum output target length 2. Beam Size for inference strategy.

2.3 Datasets

For our experiments and discussion, we use Gigawords and Amazon Fine Reviews dataset. Though we were able to package the CNN/Daily mail dataset for both the models, due to constraints of training run time, GPU memory and validation run time, we were not successful in testing our parameters with the CNN/Dailymail dataset. Our experimentation section, results and discussion will be restricted to 10K Gigawords and 500K Amazon, trained on $\approx$400K points on Model-A and Model-B.

3 Experiment Method & Fine Tuning Parameters

To compare different test cases using the ROUGE F-score, a single python script from pyrouge was used[6]. In our experiment method, we first setup the model and matched the reported values, before fine-tuning the parameters we want to test. The goal was to adapt these models for transfer learning applications, perform fine-tune training when large training data is not available and to understand parameters that can be tuned to get good summaries. We tested both models on the Gigawords dataset and compared the fine-tuned results with the authors. In addition, we cleaned and formatted the Amazon Reviews dataset for the two models, which have about 5 to 10 sentences, that is summarized to short single line sentences of 4 to 8 words. For this application we choose to fine-tune the model to test the effect of maximum input sequence length, maximum target output length, and the beam size.

- Step-1: Setup Python, Pytorch environment with dependencies to run Model-A(UNILM).
    1. Completed model setup to test predictions using the author provided model checkpoint and test dataset of 10K Gigawords to match the ROUGE scores reported by the authors.
    2. Cleaned and prepared the text data to train to 10 epochs on two datasets (Amazon Review and Gigawords10K). We limited to 10 epochs, due to computing resources and time(each epoch trains for $\approx 6-7$ hours on Nvidia GeForce RTX 2080 in BLU9402).
- Step-2: Setup Python and Tensorflow environment with dependencies to run Model-B(Lee's Model[2]).
    1. Cleaned and prepared the Amazon Reviews data to train to 10 epochs on a training set.
    2. For the Gigawords dataset, we used the author provided checkpoint for the encoder and fine-tuned for the decoder and inference strategy.
- Step-3: Compared model-A(UNILM) using our fine-tune/trained results from epoch 10 and author provided checkpoint trained for 30 epochs for Gigawords 10K. The test cases are shown in Table 1.
- Step -4: Compared model-A and model-B for the Gigawords dataset for different fine-tuning parameters. The test cases are shown in Table 2.
- Step-5: Compared model-A and model-B using our trained and saved model for the prediction on Amazon Reviews dataset for different fine-tuning parameters. The test cases are shown in Table 4. The epoch reported is based on the minimum running loss observed.

4 Results & Discussion

4.1 Step-1: Validate Baseline Model-A and Setup Model-A for Training

After we setup Model-A to run in Pytorch, to benchmark the model baseline parameters, we used a pretrained saved model checkpoint to generate predictions on the validation set. This was compared

Table 1: Step-3 Model A bench marking, hyper parameter fine-tuning, and results for Gigawords.

Test Case	Model	Checkpoint Weights	Maximum Sequence Length	Maximum Target Length	Beam Size	Rouge1	Rouge2	RougeL
REF	Model-A	Author's	192	32	5	0.2019	0.07348	0.18964
1	Model-A	Author's	192	32	5	0.2078	0.07990	0.19492
2	Model-A	Author's	192	32	10	0.2044	0.07765	0.19223
3	Model-A	Author's	128	32	5	0.2078	0.07990	0.19491
4	Model-A	Author's	128	32	10	0.2043	0.07765	0.19213
5	Model-A	Epoch 10	192	32	5	0.1031	0.02003	0.09364
6	Model-A	Epoch 10	128	32	5	0.1031	0.02003	0.09368

Table 2: Step-4 experiments table and results comparing Model-A and Model-B-Gigawords Datset

Test Case	Model	Checkpoint, Weights	Maximum Sequence Length	Maximum Target Length	Beam Size	Rouge1	Rouge2	RougeL
1	Model-A	Author's	192	32	5	0.2078	0.0799	0.19492
3	Model-A	Author's	128	32	5	0.2078	0.0799	0.19492
7	Model-B	Author's	50	5	5	0.37735	0.17041	0.35965
8	Model-B	Epoch 10	50	10	5	0.43575	0.20266	0.40913

to the predictions generated by the authors on the same dataset. The ROUGE-F scores were calculated using a python script[6], on both the author's output files and our predictions. Table 1: REF and Test Case 1 show the results for the baseline model parameters. Though the ROUGE scores are different from the values reported in literature[1](R1:0.3421, R2:0.1528 RL:0.3154), our comparison of the ROUGE scores for baseline and our predictions are similar(Baseline: R1:0.2019, R2:0.07348, RL: 0.18964, Predictions: R1: 0.2078, R2: 0.07990, RL: 0.1949). After research to investigate the difference in the scores, we discovered the scores are different from that reported in the literature, due to the difference in the Python script used to generate the ROUGE scores.

The Gigawords10K training set referenced by the authors, was also setup to train to compare our model checkpoint after each epoch using the ROUGE scores with the authors results. Since language generation tasks are unsupervised learning, they do not have the same training loss as the final objective function after decoding and output sequence generation, ROUGE scores are calculated after inference to check model performance as discussed in the introduction section. Once baseline Model-A was setup to run training, we processed the Amazon Reviews dataset and setup the model to train for 10 epochs using the same parameters used by the authors for comparison.

4.2 Step-2: Validated Baseline Model-B and Setup Model-B for Training

Similar to Step-1, we setup Model-B, and checked the ROUGE score to benchmark our model parameters. The Amazon dataset was setup to train on the Model-B for 10 epochs. We did not setup training on the Gigawords dataset for Model-B. From our observation in Step-1, we noticed that to match the authors provide model checkpoint and results, several epochs of training(30 epochs) on more than 25M datapoints was required, we chose to use the author provided model checkpoint for fine-tuning the Decoder and inference strategy, while using a pre-trained saved checkpoint model.

4.3 Step-3: Comparison of Our Trained Model-A with a Pre-Trained Model-A on the Gigawords Dataset

Table 1 shows a summary of the results from our experiments using Model A with the Gigawords dataset. The first observation is the ROUGE score from our saved model checkpoint at epoch#10, on the same training set is much lower than the author saved model checkpoint. We may need to train our model further to about 30 epochs and save the model with the best loss after about 30 epochs to match the results of the authors. Due to constrains in extensive training required for our

Table 3: Step-4 results comparing summaries created using Model-A & Model-B

Header	Text
	Model-A
Review:	police arrested five anti - nuclear protesters th ##urs ##day after they sought to disrupt loading of a f ##rench ant ##ar ##ctic research and supply vessel , a spokesman for the protesters said.factory orders for manufactured goods rose 1. 1 percent in se ##pt ##em ##ber , the commerce department said here th ##urs ##day
Summary:	protesters target f ##rench research ship
Predictions	
Test Case-1:	PA spokesman for the French Ministry of Defence said the arrest was " unprofession " and " unjust " . In response to the arrest ,
Test Case-3:	A spokesman for the French Ministry of Defence said the arrest was " unprofession " and " unjust " . In response to the arrest ,
	Model-B
Review:	hong kong share prices closed #.## percent higher friday on a rebound after thursday 's falls , with the strong response to bank of communication 's initial public offering -lrb- ipo -rrb- supporting sentiment , dealers said
Summary:	hong kong shares close higher on rebound after <unk> ipo jobs data
Predictions	
Test Case-7:	hong kong share prices close
Test Case-8:	hong kong stocks close # pct higher

Table 4: Step-5 experiments table and results comparing Model-A and Model-B-Amazon Reviews Dataset

Test Case	Model	Checkpoint, Weights	Maximum Sequence Length	Maximum Target Length	Beam Size	Rouge1	Rouge2	RougeL
9	Model-A	Epoch 4	96	32	5	0.04604	0.00699	0.04379
10	Model-A	Epoch 4	32	12	5	0.05603	0.00831	0.05389
11	Model-A	Epoch 4	24	8	5	0.05312	0.00823	0.05212
12	Model-B	Epoch 10	50	2	5	0.14707	0.04878	0.14649
13	Model-B	Epoch 10	50	10	5	0.16491	0.05194	0.16347

encoder transformers starting with the Bert Baseline, we limited our fine-tuning to the decoder and inference parameters, while using the authors saved model checkpoint presented in further discussion. The parameters we changed are: 1. beam size, and 2 maximum input sequence length. The results in Table 1 show that for the Gigawords dataset, the changes in beam size or the maximum input sequence length did not affect the ROUGE scores or improve the quality of our summaries. Several authors have reported that fine tuning beam size will result in better predictions[2][8], this could be the case in large input text(more than 15 sentences) and summaries that are several sentences long(about 3 to 5 sentences) similar to the CNN/dailymail dataset.

4.4 Step-4: Comparison of Model-A & Model-B on the Gigawords Dataset

In this section, we compare the results from Model-A and Model-B on the Gigawords dataset Table2. We use our learning from Step-3, and the beam size is kept constant. Pre-trained Model-B had better summaries than Model-A. The pre-trained Model-B used was trained on a larger number of datapoints(400K) compared to 10K datapoints used in Model-A, this could have resulted in a better ROUGE score for Model-B compared to Model-A. The python script reported by author's of Model-A to calculate ROUGE score is different from that used by author of Model-B, we are using the script presented by Model-B to maintain consistency and comparability. It was surprising to observe that Model-A which claims to be a advanced text summarization model with better ROGUE metric(using the same author fine-tuned parameters reported), results in a lower score when tested using a different

Table 5: Step-5 results comparing summaries created using Model-A & Model-B

Header	Text
	Model-A & Model-B
Review:	great sugar free option peanut butter fans cutting sugar one better carb options skippy discontinued comes creamy
Summary:	great sugar free option
Predictions	
Test Case-9:	Peanut butter fans cut sugar free peanut butter fan cut carb options skippy , skippy and skippy creamy . Peanut cream fans cut car
Test Case-10:	peanut butter fans cutting sugar one better carb option skip
Test Case-11:	peanut butter fans cut sugar one better
Test Case-12:	peanut butter
Test Case-13:	guiltless indulgence
	Model-A & Model-B
Review:	marinade great hot mild right price excellent compared buying single bottles grocery store delivery time good overall great product enhance eating experience par jamaica jerk recipes restaurants may experienced fire breathing spice right kick chicken pork ribs
Summary:	great spicy marinade for chicken ribs pork and wings
Predictions	
Test Case-9:	marinade marinate margade margate magar magar and magar gravy magar . marina marga maravy
Test Case-10:	marinade excellent compared to single bottles grocery store delivery time
Test Case-11:	marinade great hot mild marina
Test Case-12:	marinade great
Test Case-13:	salad dressing at its best

ROUGE python script. Table 3 shows example text summaries predicted by Model-A & Model-B, as shown in the examples the summaries produced by both the models are different and fairly good in its own respect. Model-A, produces longer summaries due to a longer maximum output target sequence length, while Model-B produces shorter summaries. A small reduction in the maximum target length for the decoder in Model-A could result in a better ROUGE score for Model-A using this ROUGE python script[6].

4.5 Step5: Comparison of Model-A & Model-B on the Amazon Reviews Dataset.

Model-A and Model-B were trained and tested using the same dataset. The fine-tuning parameter changed was the maximum output target sequence length. Table 4 shows the results for the two models, we noticed Model-B has a higher ROUGE score compared to Model-A, though the later was expected to have better results for this task. This again could be due to limited fine-tuning training carried out using Model-A for this dataset. Table 5 shows the predicted summary for the same input text for different maximum output sequence lengths using Model-A and Model-B. Even with limited training on UNILM, we are able to get reasonable good summaries. We may be able to train Model-A to larger number of epochs, in this case we are using Epoch4 results, the model could potentially achieve better results after training to about 20 to 30 epochs and along with parameter fine-tuning that gives best results.

5 Conclusion & Learning:

- For transfer learning applications of abstractive text summarization using UNILM model or the bidirectional stacked LSTM model, fine-tuning training requires several epochs (20 to 30). More the training data available, better is the model performance to generate summaries with good ROUGE scores.
- Fine-tuning maximum input sequence length and beam size did not make much difference for small summaries in the Gigawords and Amazon Reviews dataset. Fine-tuning the maxi-

mum output target length, produced predictions that compared well with target summaries provided in the dataset.

- We expected the UNILM model to have better predictions, however from our observations, the three stacked bidirectional LSTM model had better performance. Due to limitations in time and computation resource, we performed limited training on the UNILM model, better results could be achieved by fine-tuning parameters and extended training.
- ROUGE scores have significant variations. It is difficult to have a standardized comparison metric for abstractive text summarization. Different scripts are available to calculate the scores, while the scores alone do not accurately predict model performance.
- For transfer learning applications, in addition to the ROUGE scores, there are limitations to evaluate the quality of the summary to compare and estimate model performance.

6 Contributions

- **Arjun Mahadevan 20%:** Project planning and literature review. Setup UNILM model to run in Pytorch. Worked on Step-1, Step-3 & Step-4. Prepared poster on LaTeX. Report writing(Sections: Experiments & Methods, Results& Discussion, Conclusion).
- **Anchal Jain 20%:** Literature review. Setup Model-B in tensorflow and tested for Gigawords. Worked on Step-2, Step-4 and Step-5. Setup ROUGE score script and generated scores and summary results. Tables for the poster. Report writing(Sections: Introduction, Result Tables, Results&Discussion).
- **Ogheneovo Ojameruaye 20%:** Literature review, setup Model-B in tensorflow. Worked on training the Amazon Review Dataset for Model-B in tensorflow. Worked on Step -2, Step-4 and Step-5. Generated summary results tables for the poster. Report writing(Sections:Abstract, Results Tables, Results&Discussion).
- **Nirosha Bugatha 20%:** Setup UNILM model to run in Pytorch. Data cleaning and preparation of text data for training in Model-A, and Model-B. Worked on Step-1, Step-2 & Step-5. Poster preparation (Generated figures, literature review and results). Report writing(Sections: Literature Review, Results and Discussion).
- **Shahram Akbarinasaji 20%:** Literature review. Setup Model-B in tensorflow. Cleaned and prepared dataset for training. Worked on Step-2 and Step-5 to run different iterations for the Amazon Dataset. Prepared table templates for the poster. Report Writing (Introduction, Model Architecture & Dataset).

References

[1] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. *https://github.com/microsoft/unilm*, abs/1905.03197, 2019.

[2] Dongjun Lee. Simple tensorflow implementation of text summarization using seq2seq library. *GitHub repository-https://github.com/dongjun-Lee/text-summarization-tensorflow*, 2018.

[3] C. Y. Lin. Rouge: A package for automatic evaluation of summaries. *ACL Text Summarization Workshop*, 2004.

[4] Elena Lloret, Laura Plaza, and Ahmet. The challenging task of summary evaluation: An overview. *Journal of language resources and Evaluation*, Vol. 52, 2018.

[5] K. Papineni, S. Roukos, T. Ward, and W. Zhu. Bleu: A method for automatic evaluation of machine translation. *Proceedings of ACL*, 2002.

[6] Pltrdy. Files2rouge. *GitHub repository-https://github.com/pltrdy/files2rouge*.

[7] Sebastian Ruder. Nlp progress text summarization, 2019.

[8] Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks, 2017.