**Text Summarization**

Ujwal Tandon Shubham Rohatgi Gautam Gupta Satvika Thakur

[ujwaltandon04@gmail.com](mailto:ujwaltandon04@gmail.com) [srohatgi440v@gmail.com](mailto:srohatgi440v@gmail.com) [gautamgupta94@gmail.com](mailto:gautamgupta94@gmail.com) [satvikathakur@gmail.com](mailto:satvikathakur@gmail.com)

**Abstract**

Automatic text summarization is basically summarizing of the given paragraph using natural language processing and machine learning. There has been an explosion in the amount of text data from a variety of sources. This volume of text is an invaluable source of information and knowledge which needs to be effectively summarized to be useful. In this review, the main approaches to automatic text summarization are described. We review the different processes for summarization and describe the effectiveness and shortcomings of the different methods. Two types will be used i.e.-extractive approach and abstractive approach. The basic idea behind summarization is finding the subset of the data which contains the information of all the set. There is a great need to reduce unnecessary data. It is very difficult to summarize the document manually so there is the great need of automatic methods. Approaches have been proposed inspired by the application of deep learning methods for automatic machine translation, specifically by framing the problem of text summarization as a sequence-to-sequence learning problem.

**1 Introduction**

In the modern Internet age, textual data is ever increasing. Need some way to condense this data while preserving the information and meaning. We need to summarize textual data for that. Text summarization is the process of automatically generating natural language summaries from an input document while retaining the important points. It would help in easy and fast retrieval of information.

There are two prominent types of summarization algorithms.

• **Extractive summarization systems** form summaries by copying parts of the source text through some measure of importance and then combine those part/sentences together to render a summary. Importance of sentence is based on linguistic and statistical features.

• **Abstractive summarization systems** generate new phrases, possibly rephrasing or using words that were not in the original text. Naturally abstractive approaches are harder. For perfect abstractive summary, the model has to first truly understand the document and then try to express that understanding in short possibly using new words and phrases. Much harder than extractive. Has complex capabilities like generalization, paraphrasing and incorporating real-world knowledge.

Majority of the work has traditionally focussed on extractive approaches due to the easy of defining

hard-coded rules to select important sentences than generate new ones. Also, it promises grammatically correct and coherent summary. But they often don’t summarize long and complex texts well as they are very restrictive.

**Overview of the final process**

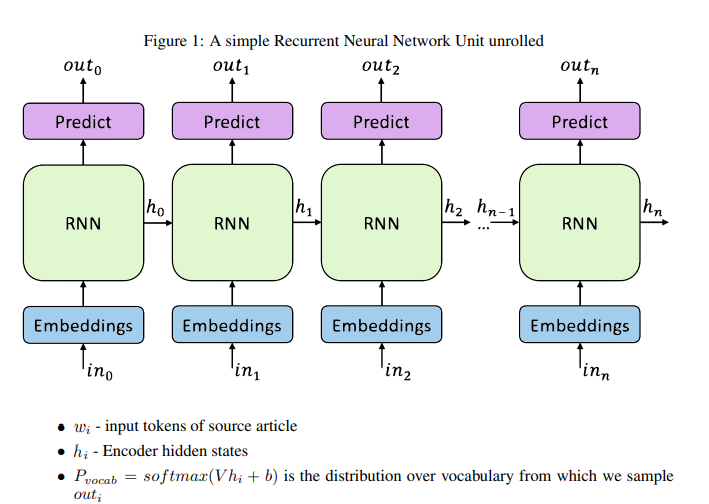
The traditional rule-based AI did poorly on Abstractive Text Summarization. But the recent advances in Deep Learning changed that for the good. Inspired by the performance of Neural Attention Model in the closely related task of Machine Translation Rush et al. [2015] applied this Neural Attention Model to Abstractive Text Summarization and found that it already performed very well and beat the previous non Deep Learning-based approaches. But they applied this only to single sentences and didn’t generalize. Then Nallapati et al. [2016] generalized this further and set the baseline Model. We explore this model in detail and then see it’s shortcomings. Further improvements have been done to this baseline model by See et al. [2017] using Pointer Generator Networks and Coverage Mechanism by **Abigail See** approach significantly improve the ROUGE scores over the baseline.

**3 Baseline Neural Attention Model**

The Neural Attention Model as introduced by Bahdanau et al. [2014] consists of an encoder-decoder RNN with attention Mechanism. The inputs that we feed into the RNN are word/morpheme/phrases embeddings. We use word embeddings for summarization.

3.1 Simple Encoder-Decoder Model

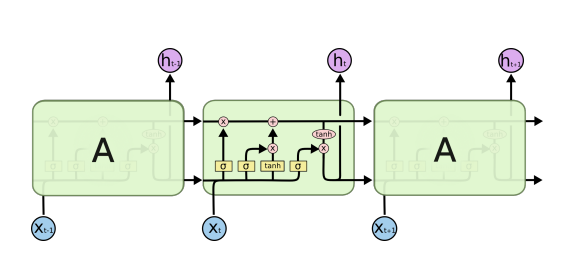
.



There are many variants of RNN’s like LSTM’s and GRU’s and NAS. A detailed view of an LSTM unit with all of its layers is described below. The motivation behind using an LSTM is that it captures long-term dependency pretty well and the information in the starting of the sequence is able to traverse down the line. This is done by selectively selecting and restricting the flow of information in the LSTM unit. There are three gates in an LSTM. Forget Gate Layer

• ft = σ(Wf [ht−1, xt] + bf )

• Ct = Ct ⊗ f



Input Gate Layer 1

• it = σ(Wt[ht−1, xt] + bi)

• Cft = tanh(Wi [ht−1, xt] + bc)

• Ct = Cft ⊗ tt + C

t Output Gate Layer

• ot = σ(Wo[ht−1, xt] + bo)

• ht = ot ∗ tanh(Ct)

We also use Bidirectional RNN’s which scan the input from both left and right. This done so that at any step both the words to the right and words to the left are included in the context. • We make two passes on the source sequence, computing hidden states for backward pass ←−ht and for forward pass −→ht .

**3.2 Attention Mechanism**

The basic encoder-decoder model performs okay on very short sentences but it fails to scale up.

• The main bottleneck is the fixed sized encoding of the input string, which is the LSTM output of the last time-step. Due to it’s fixed size it is not able to capture all the relevant information of the input sequence as the model sizes up.

• At each generation step, only a part of the input is relevant. So how does the model decide which part of the input encoding to focus on at each generation step?

• At each step, the decoder outputs hidden state, from which we generate the output. The solution to this problem is using the attention model. This model calculates the importance of each input encoding for the current step by doing a similarity check between decoder output at this step and input encodings.

Doing this for all of the input encodings and normalizing, we get an importance Vector. We then convert it to probabilities by passing through softmax. Then we form a context vector by multiplying with the encodings.

• importanceit = V ∗ tanh(eiW1 + htW2 + battn).

• Attention Distribution a t = softmax(importanceit)

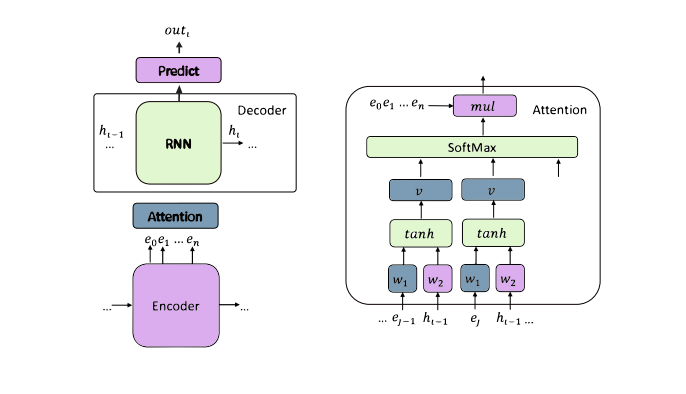
• Context Vector h ∗ t = P i ei ∗ a t i

2 . Context Vector is then fed into two layers to generate distribution over the vocabulary from which we sample.

• Pvocab(w) = softmax(V 0 (V [ht, h∗ t ] + b) + b 0 )

• For the loss at time step t, lost = − log P(w ∗ t ), where w ∗ t is the target summary word.

. • We then use the Backpropagation through time Algorithm to get the gradients and then apply any of the popular Gradient Descent algorithms to minimize the loss and learn good parameters.



# **Describe the dataset**

Hermann et al. (2015) created two awesome datasets using news articles for Q&A research. Each dataset contains many documents (90k and 197k each), and each document companies on average 4 questions approximately. Each question is a sentence with one missing word/phrase which can be found from the accompanying document/context.

## CNN

* Questions: [here](https://drive.google.com/uc?export=download&id=0BwmD_VLjROrfTTljRDVZMFJnVWM)
* Stories: [here](https://drive.google.com/uc?export=download&id=0BwmD_VLjROrfTHk4NFg2SndKcjQ)
* Raw HTML: [here](https://nyu.box.com/s/m1iw24cudp1l99krnj46qz97uhbecvvf)

This dataset contains the documents and accompanying questions from the news articles of CNN. There are approximately 90k documents and 380k questions. Available 'questions/', which should be sufficient to reproduce the setting from the original paper, and 'stories/', which can be useful for other uses of this dataset.

## Daily Mail

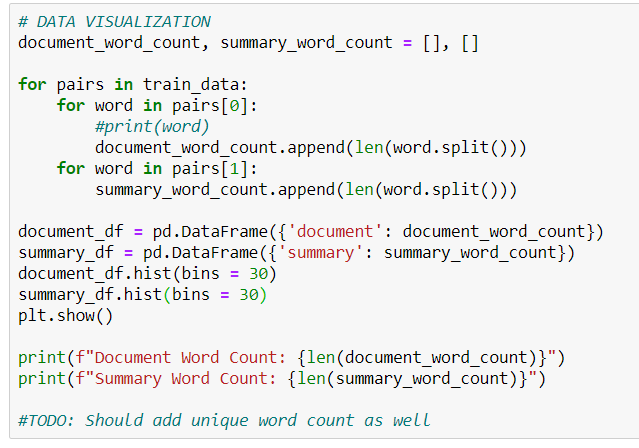
* Questions: [here](https://drive.google.com/uc?export=download&id=0BwmD_VLjROrfN0xhTDVteGQ3eG8)
* Stories: [here](https://drive.google.com/uc?export=download&id=0BwmD_VLjROrfM1BxdkxVaTY2bWs)
* Raw HTML: [here](https://nyu.box.com/s/derkw9ujca8bx38stuy8jo1d76ar1r6f)

This dataset contains the documents and accompanying questions from the news articles of Daily Mail. There are approximately 197k documents and 879k questions. Available 'questions/', which should be sufficient to reproduce the setting from the original paper, and 'stories/', which can be useful for other uses of this dataset.

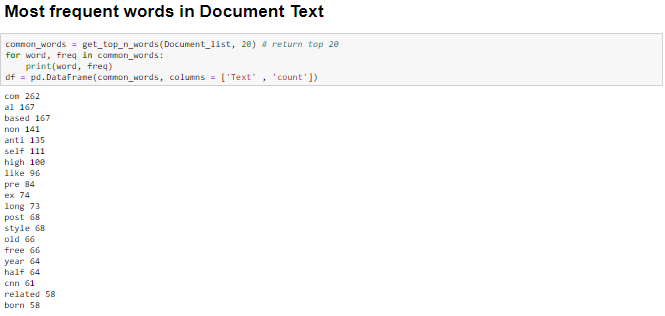
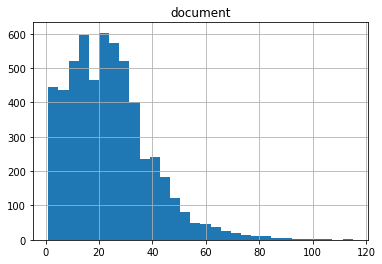
**Exploratory Data Analysis**

Exploratory Data Analysis (EDA) is been done on the sample size of 2000 documents for CNN /Daily Mail dataset to understand better. Document and summery is been converted into two separate list of words then we have set the data pipeline for per-processing .

we used nltk library then followed lemmatize, punctuation, Tags, stop words and then tokenize, that control the number of sample size to be evaluated.



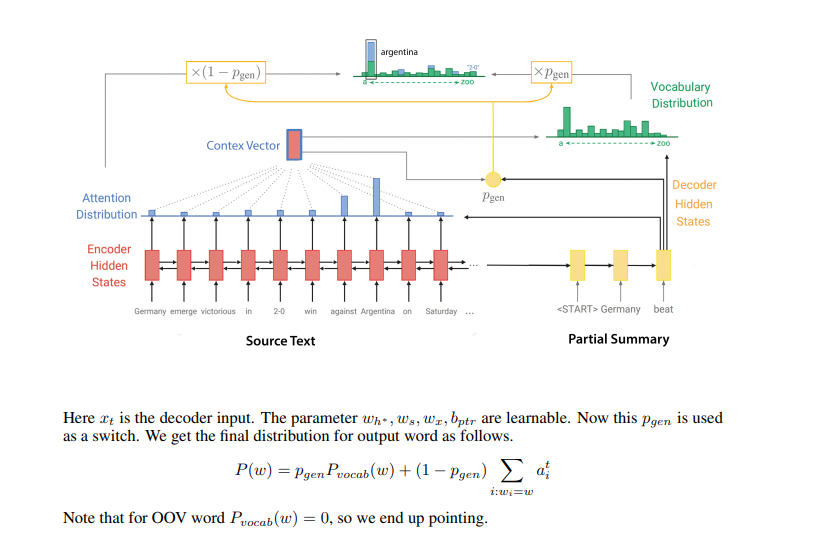
Below is the graph of total word count in each document with majority documents having count between 400 to 600



**Step-by-step walk through of the solution**

Finally we selected following model to implemented CCN daily mail

**Pointer Generator Network**

Introduced by Nallapati et al. [2016] and See et al. [2017]. It helps to solve the challenge of OOV words and factual errors.It works better for multi-sentence summaries. The basic idea is to choose between generating a word from the fixed vocabulary or copying one from the source document at each step of the generation. It brings in the power of extractive methods by pointing (Vinyals et al. [2015]). So for OOV words, simple generation would result in UNK, but here the network will copy the OOV from source text. At each step we calculate generation probability pgen = σ(w T h∗h ∗ t + w T s ht + w T x xt + bptr)

**Coverage Mechanism**

This concept was first applied by See et al. [2017]. The cause of repetitiveness of the model can be accounted for by increased and continuous attention to a particular word. So we can use Coverage Model by Tu et al. [2016]. We form a Coverage vector as follows,



Intuitively, by summing the attention at all steps we are keeping track of how much coverage each encoding, e i has received. Now, give this as input to attention mechanism. The updated formulate for calculating importance is,



Now to make the network learn to not focus on things that have already been covered we penalize attending to things that have already been covered



This loss penalizes overlap between attention at this step and coverage till now. Final loss is,



**Execution steps**

**### Get the dataset**

To obtain the CNN / Daily Mail dataset, follow the instructions [here](https://github.com/abisee/cnn-dailymail). Once finished, you should have [chunked](https://github.com/abisee/cnn-dailymail/issues/3) datafiles `train\_000.bin`, ..., `train\_287.bin`, `val\_000.bin`, ..., `val\_013.bin`, `test\_000.bin`, ..., `test\_011.bin` (each contains 1000 examples) and a vocabulary file `vocab`.

**### Run training**

To train your model, run:

python run\_summarization.py --mode=train --data\_path=/path/to/chunked/train\_\* --vocab\_path=/path/to/vocab --log\_root=/path/to/a/log/directory --exp\_name=myexperiment

This will create a subdirectory of your specified `log\_root` called `myexperiment` where all checkpoints and other data will be saved. Then the model will start training using the `train\_\*.bin` files as training data.

\*\*Increasing sequence length during training\*\*: Note that to obtain the results described in the paper, we increase the values of `max\_enc\_steps` and `max\_dec\_steps` in stages throughout training (mostly so we can perform quicker iterations during early stages of training). If you wish to do the same, start with small values of `max\_enc\_steps` and `max\_dec\_steps`, then interrupt and restart the job with larger values when you want to increase them.

**### Run (concurrent) eval**

You may want to run a concurrent evaluation job, that runs your model on the validation set and logs the loss. To do this, run:

```

python run\_summarization.py --mode=eval --data\_path=/path/to/chunked/val\_\* --vocab\_path=/path/to/vocab --log\_root=/path/to/a/log/directory --exp\_name=myexperiment

```

Note: you want to run the above command using the same settings you entered for your training job.

**\*\*Restoring snapshots\***\*: The eval job saves a snapshot of the model that scored the lowest loss on the validation data so far. You may want to restore one of these "best models", e.g. if your training job has overfit, or if the training checkpoint has become corrupted by NaN values. To do this, run your train command plus the `--restore\_best\_model=1` flag. This will copy the best model in the eval directory to the train directory. Then run the usual train command again.

**### Run beam search decoding**

To run beam search decoding:

python run\_summarization.py --mode=decode --data\_path=/path/to/chunked/val\_\* --vocab\_path=/path/to/vocab --log\_root=/path/to/a/log/directory --exp\_name=myexperiment

Note: you want to run the above command using the same settings you entered for your training job (plus any decode mode specific flags like `beam\_size`).

This will repeatedly load random examples from your specified datafile and generate a summary using beam search. The results will be printed to screen.

\*\*Visualize your output\*\*: Additionally, the decode job produces a file called `attn\_vis\_data.json`. This file provides the data necessary for an in-browser visualization tool that allows you to view the attention distributions projected onto the text. To use the visualizer, follow the instructions [here](https://github.com/abisee/attn\_vis).

If you want to run evaluation on the entire validation or test set and get ROUGE scores, set the flag `single\_pass=1`. This will go through the entire dataset in order, writing the generated summaries to file, and then run evaluation using [pyrouge](https://pypi.python.org/pypi/pyrouge). (Note this will \*not\* produce the `attn\_vis\_data.json` files for the attention visualizer).

**### Evaluate with ROUGE**

`decode.py` uses the Python package [`pyrouge`](https://pypi.python.org/pypi/pyrouge) to run ROUGE evaluation. `pyrouge` provides an easier-to-use interface for the official Perl ROUGE package, which you must install for `pyrouge` to work. Here are some useful instructions on how to do this:

\* [How to setup Perl ROUGE](http://kavita-ganesan.com/rouge-howto)

\* [More details about plugins for Perl ROUGE](http://www.summarizerman.com/post/42675198985/figuring-out-rouge)

**Model evaluation**

We need to evaluate the quality and coherence of the summary. By quality, we mean how well does the summary capture the source document and by coherence we mean if it’s grammatically correct and is human readable. In the dataset, for a given document we may or may not be given a golden target summary.

• If the target summary is not given, then we need to make a similarity measure between the summary and the source document. The crux is that in a good summary, the topics covered would be roughly the same as that of the topics covered in the document. So we may use topic models like Latent Semantic Analysis(LSA) and Latent Dirichlet Allocation(LDA) to measure the similarity between topics.

• If the target summary is given then it is better to use metrics like ROUGE and METEOR. They are essentially string matching metrics. ROUGE is the most popular and has various variants like ROUGE-N and ROUGE-L. ROUGE-N measures the overlap of N-grams between the system and reference summary. A bigger N implies more fluency in the summary, because matching with a bigger portion of the reference summary which is more fluent, implies more fluency. ROUGE-L is based on longest common subsequence, thereby also taking into account sentence level similarity. ROUGE-S is the skip-gram variant which though not very popular currently but does seem promising to use for longer summaries

**Model Outcome(Example)**

|  |
| --- |
| ARTICLE: a super slimmer who swelled to 26 stone after eating a loaf of bread a day is now toasting her diet - and literally becoming half the woman she used to be . michelle quinn , 42 , ballooned after munching her way through toast , \_sarnies\_ and slices of bread all day . her diet used to be made up of white toast with margarine for breakfast , sandwiches and crisps for lunch and fish and chips and takeaways for dinner . michelle quinn has lost half of her body weight after ditching her \_bread-based\_ diet which saw her eat a loaf a day , she has also dropped from a clothes size 30 -lrb- left -rrb- to a size 12 -lrb- right -rrb- . before losing weight 43-year-old michelle was a size 30 , here she proudly holds up a pair of her old trousers . but she gave up the bread and started a diet of breakfast of cereal or fruit and yoghurt , home-made soup for lunch and healthy versions of her favourite meals . michelle , of south shields , tyneside , says she feels like a new woman after losing 12.5 st and dropping from dress size 30 to size 12 . she has been named slimming world 's greatest loser in the west \_harton\_ area of south shields . she said : ' i feel like a new woman since losing weight . in fact , i look so different that people who i \_havent\_ seen for a while often ca n't believe i 'm the same person . ` for me though it 's the change on the inside that 's been the biggest - i 'm happier , healthier and much more confident now . michelle , pictured with her uncle derek , joined a slimming group in 2013 in a bid to shift the weight , she says she had struggled with high blood pressure , back pain and that she got breathless easily . michelle was not fat as a child but piled on the pounds thanks to her diet of fish and chips and \_sandwhiches\_ . now a size 12 , michelle no longer feels the need to eat a whole loaf of bread every day . she continued : ' i still enjoy all my favourite meals like burgers and chips and roast dinners but i 've learned how to make small changes like using lean meat or cooking with low calorie spray instead of oil or butter . ` it fits in really well with the rest of my family and we can all eat the same meals . ' michelle weighed \_25st\_ 3lbs when she joined the group in april 2013 and has since dropped to 12st 10lbs . she said : ` before i lost the weight i hid behind a big bubbly personality . i 'd pretend it did n't bother me that i was bigger than most other people , but that was far from the truth . ' i hated shopping for clothes and found just climbing up stairs and doing simple everyday tasks would leave me tired and out of breath . ' michelle 's weight was also putting a huge strain on her health and she suffered with high blood pressure , chronic back pain and got breathless easily. |
| **Reference Text**: michelle quinn ballooned thanks to her addiction to bread and chips . the 42-year-old would eat a whole loaf of bread a day . in 2013 she weighed more than !!\_25st\_!! and was a size 30 dress size . after joining a slimming group her weight has dropped to 12 stone 10lbs . |
| **Decoded Text:** michelle quinn , 42 , ballooned after munching her way through toast , sarnies and slices of bread all day . her diet used to be made up of white toast with margarine for breakfast , sandwiches and crisps for lunch and fish and chips for dinner . before losing weight 43-year-old michelle was a size 30 , here she proudly holds up a pair of her old trousers . |

Infrastructure used :- Local Linux System RAM: 16 GB GPU: GTX 950 (2 GB memory) Processor: 4 CPUs Tensorflow 1.14 is being used for our model

**Note -Difficult to setup CUDA on Windows machine , Pyrouge should be updated And Wordnet should be updated & aligned with Pyrouge**

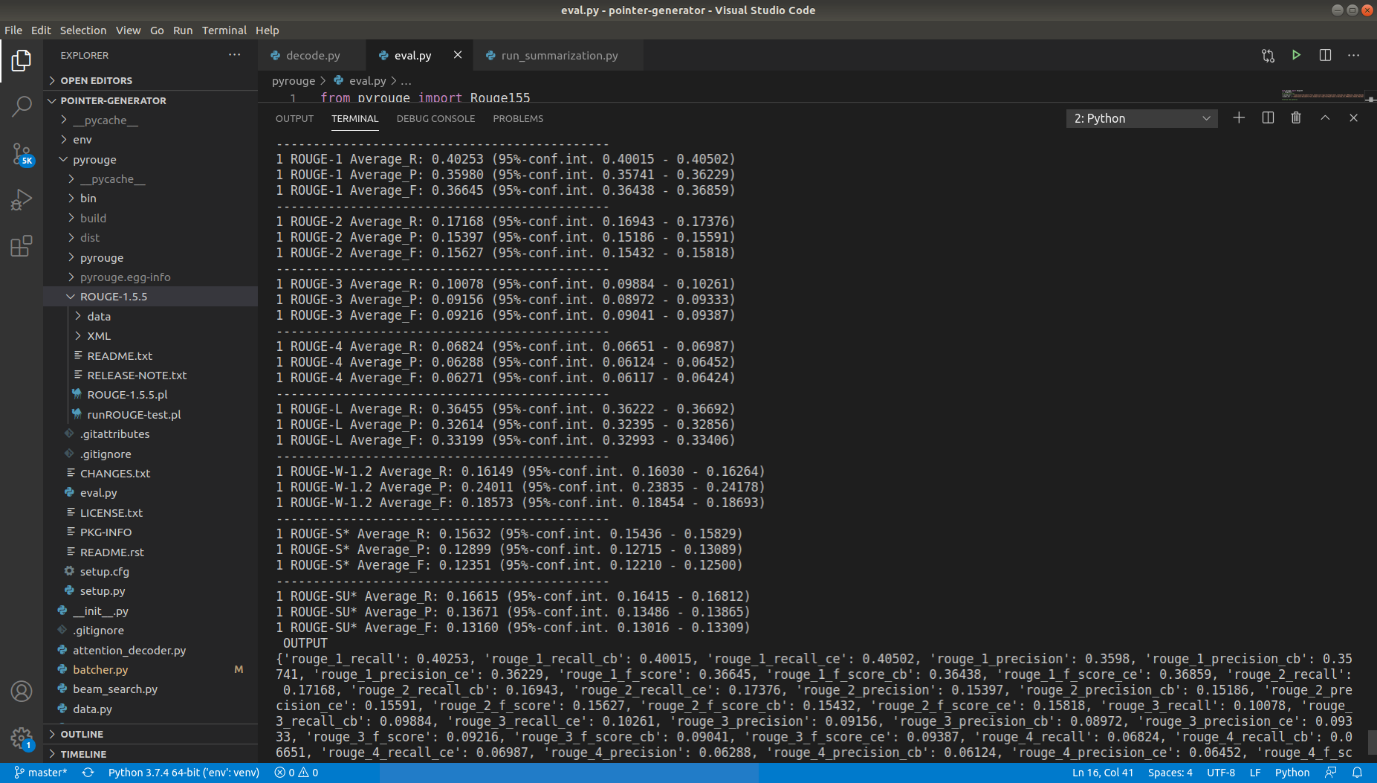
**Actual model result**

Rouge Score for Pointer Generator without Coverage

|  |  |  |  |
| --- | --- | --- | --- |
| **Without Coverage** | | | |
| **Column1** | **F Score** | **Precision** | **Recall** |
| ROGUE-1 | 0.36645 | 0.3598 | 0.40253 |
| ROGUE-2 | 0.15627 | 0.15397 | 0.17168 |
| ROGUE-3 | 0.09216 | 0.09156 | 0.10078 |
| ROGUE-4 | 0.06271 | 0.06288 | 0.06824 |
| ROGUE-L | 0.33199 | 0.32614 | 0.36455 |
| ROGUE-W 1.2 | 0.18573 | 0.24011 | 0.16149 |
| ROGUE-S\* | 0.12351 | 0.12899 | 0.15632 |
| ROGUE-SU\* | 0.1316 | 0.13671 | 0.16615 |

**Rouge Score for Pointer Generator with Coverage**

|  |  |  |  |
| --- | --- | --- | --- |
| **With Coverage** | | | |
| **Column1** | **F Score** | **Precision** | **Recall** |
| ROGUE-1 | 0.39412 | 0.38598 | 0.4365 |
| ROGUE-2 | 0.16967 | 0.16701 | 0.18746 |
| ROGUE-3 | 0.09966 | 0.09901 | 0.10942 |
| ROGUE-4 | 0.06739 | 0.06758 | 0.07366 |
| ROGUE-L | 0.3589 | 0.35174 | 0.39724 |
| ROGUE-W 1.2 | 0.19911 | 0.25635 | 0.17444 |
| ROGUE-S\* | 0.13472 | 0.14065 | 0.17314 |
| ROGUE-SU\* | 0.14325 | 0.14875 | 0.1837 |



**Limitations and Problem**

Though the baseline gives decent results, they are clearly plagued by many problems

1. They sometimes tend to reproduce factually incorrect details.

2. They struggle with Out of Vocabulary (OOV) words. So we see many UNK tokens in the summary.

3. They are also a bit repetitive and focus on a word/phrase multiple time.

4. Focus is mainly on single sentence summary tasks like headline generation.

**Closing Reflections**

A majority of the dataset that is available online is news dataset. A peculiar characteristic of these news datasets is that one can come up with a pretty good summary only by looking at the top few sentences. All the above-discussed models discussed above assume this and look at only the top 5-6 sentences of the source article. We need a richer dataset for multi-sentence Text Summarization. Also, another peculiarity of these models as with any other Deep Learning model is that they require huge amounts of data and computational power.

**Problem with the metric ROUGE as a Metric is deficient.**

Pointer Generator Networks and Coverage Mechanism by **Abigail See**, it is possible to achieve a very high ROUGE score, without the summary being human readable. This clearly means that the way ROUGE measures summary is different from how humans evaluate a summary.

**Conclusion**

This review shows that Deep Learning based approaches are promising and give some hope in solving Abstractive text summarization which had been largely unsolved till now. But the problems with the metric and lack of dataset are a challenge to scalability and generalizing to multi-sentence summarization.

**References**

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014. Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al.

Abstractive text summarization using sequence-to-sequence rnns and beyond. arXiv preprint arXiv:1602.06023, 2016. Romain Paulus, Caiming Xiong, and Richard Socher.

A deep reinforced model for abstractive summarization. arXiv preprint arXiv:1705.04304, 2017. Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. arXiv preprint arXiv:1509.00685, 2015. Abigail See, Peter J Liu, and Christopher D Manning.

Get to the point: Summarization with pointergenerator networks. arXiv preprint arXiv:1704.04368, 2017. Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li.

Modeling coverage for neural machine translation. arXiv preprint arXiv:1601.04811, 2016. Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks.

.