Abstractive Summarization of WikiHow Articles

Vivek Agarwal (James Kunz's Section), Vivian Lu (Joachim Rahmfeld's Section)
December 2019

Abstract
Abstractive summarization--while still a challenging task in the NLP field--has primarily been conducted on the CNN/Daily Mail dataset [1]. Due to the reporting style of most news articles, abstractive summarization for the CNN/Daily Mail dataset typically only requires the first paragraph of the entire article to perform relatively well. This paper seeks to perform a similar abstractive summarization (headlines) on the WikiHow dataset, a task that requires longer comprehension of multiple paragraphs.

Keywords: Abstractive summarization, wikihow

Introduction

Our goal for this project was to implement an abstractive summarization model to generate headlines to summarize paragraphs within a WikiHow article. Namely, we hoped to implement a more robust version of Lopyrev's 2015 paper on generating news headlines on the CNN/Daily Mail dataset [2]. News articles typically hold their most important information within the first paragraph, and thus an abstractive summarization model does not need to read through the entire article to produce human-like summarizations. In contrast, WikiHow articles have writing styles that not only vary widely between authors but also can contain either step by step instructions or procedural essays.

Ultimately, abstractive summarization continues to be an area of active research due to its many practical applications. Examples include knowledge management, improvements in productivity, and wide-spread easier access to information for the old/visually-impaired.

Background

Lopyrev [2] primarily employs the standard encoder-decoder LSTM architecture with Bahdanau attention after restricting all CNN/Daily Mail articles to only the first paragraph. Given the differences in WikiHow, we knew that we had to consider the following potential challenges:

1. Length of WikiHow articles (detailed further within the *Data* subsection under **Approach**)
   WikiHow articles show more variation in terms of length compared to the CNN/Daily Mail dataset. We could either read the full article or read a paragraph at a time.
2. Preprocessing (detailed further within the *Data* subsection under **Approach**)
   Preprocessing varies greatly depending on the particular use case at hand (See Et Al [3] where they do no preprocessing, while common NLP tasks typically mention lower-casing, stop word removal, and punctuation clean-up).
3. Evaluation (discussed in further detail within the **Results** section)
   Most of the published papers on abstractive summarization with the CNN/Daily Mail set typically used F1-scores of ROUGE-1, ROUGE-2, and ROUGE-L [1].

Approach

Data
Our dataset is the WikiHow dataset by Koupaee and Wang [4]. It consists of over 230,000 articles, each of which contains the article's text, associated headlines (summary sentences of all paragraphs), and an associated title. There are two versions of the WikiHow Dataset: (1) WikiHowAll, where each instance is one full WikiHow article with its associated text, headlines, and title; (2) WikiHowSep, where each instance is a paragraph with its associated paragraph text, paragraph headline (summary), and article title.

We opted for WikiHowSep where each training instance is its own paragraph text and headline. This helped in scaling down our text length making it conducive for quick training and also provided us with more training samples compared to taking in full document at a time.

After a significant trial and error, we settled on the following preprocessing scheme, where all text were lowercased and tokenized, contractions were mapped, punctuation symbols were retained, and we did not use start and end tags. Lastly, pre-trained GloVe word embeddings (Wikipedia 2014 + Gigaword 5) of 300 dimensions for top 40000 words was used .

Finally, paragraphs with missing or "n/a" headlines or text were dropped. Data was then shuffled, and then split to reflect 70% training, 20% cross-validation, and 10% test split.

Code: https://github.com/vivekmids/nlp-summarization/blob/master/preparedata.py
Code: https://github.com/vivekmids/nlp-summarization/blob/master/forward/BTmGPUTF2.py

Modeling
Lopyrev mentions the basic encoder-decoder architecture as consisting of 4 hidden LSTM layers, each with 600 units. He mentions attempting dropout, which turned out to not be useful. A global attention layer is incorporated as well, with which we implement with a custom Bahdanau attention layer (https://github.com/thushv89/attention_keras). Furthermore, Lopyrev trained for only 10 epochs. Lastly, to evaluate loss, we chose to employ sparse categorical crossentropy as our labels (headlines) consisted of integers and not one-hot encoded matrices.

Code for Basic Encoder-Decoder Training:
https://github.com/vivekmids/nlp-summarization/blob/master/forward/BTmGPUTF2.py
Code for Basic Inference:
https://github.com/vivekmids/nlp-summarization/blob/master/forward/Inference.ipynb
Code for Backwards LSTM architecture:
https://github.com/vivekmids/nlp-summarization/blob/master/backwards/BTmGPUTF2.py

Results
In total, we ran 7 experiments, which are depicted below in *Table 1*:

Table 1:

No.	Units	Epochs	Dir	Optim.	Greedy Search			BEAM Search*		
					ROUGE-1-R	ROUGE-1-P	ROUGE-1-F	ROUGE-1-R	ROUGE-1-P	ROUGE-1-F
1	200	10	F	Adam	0.183072	0.262422	0.198823	0.19864	0.214122	0.183949
2	300	8	B	Adam	0.169578	0.226068	0.177968	0.149727	0.237734	0.170399
3	600	10	F	Adam	0.203247	0.266324	0.212307	0.185919	0.284992	0.208523
4	400	10	F	Adam	0.18795	0.254068	0.199056	0.173054	0.273554	0.196867
5	400	20	B	Adam	0.180064	0.241928	0.189776	0.160312	0.252536	0.18185
6	400	10	F	rmsprop	0.198505	0.251479	0.203312	0.179791	0.211557	0.18014
7	400	10	B	rmsprop	0.182827	0.248809	0.194041	0.159533	0.197426	0.163613

* Beam Search = 3 for B

A brief description of the table's columns are as follows:

- "No.": depicts experiment number
- "Units": depicts number of units in a hidden layer
- "Epochs": denotes the number of epochs run for training and cross-validation
- "Dir": denotes the direction of the LSTM layers ("F" for forwards, "B" for backwards").
- "Optim.": denotes the optimizer (Adam or RMSProp)

Various Experimentations

Our initial attempts on a single GPU instance led to memory errors when replicating Lopyrev's implementation of 600 units. Hence, our first couple runs were with small number of hidden units. Eventually, we did successfully implement a multi-GPU instance and were able to train a model similar to Lopyrev. Due to time/money restrictions, we opted for fewer hidden units (400 each for experiments 4 to 7) for quicker training.

During training, we noticed that all experiments showed a leveling off of the loss curve in our cross-validation after the 4th-5th epoch. Only experiment 5 showed a noticeable decrease in loss even after 10 epochs, with which we ran the model for 20 epochs in hopes of better performance. We acknowledge the consequences of such actions as the differences in training epochs could over/under-fit models, thus making them not fully comparable.

In summary, we experimented with the following model changes and further discussions on our observations are in the upcoming *Error Analysis* section:

1. Direction:
    a. Lopyrev experiments with forward directions in the encoder. Bahdanau et al in 2016 [5] mention the increases in performances with backwards RNNs.

2. Greedy vs BEAM search
    a. In addition to greedy search, Lopyrev mentions using a BEAM search of size 2. Our model during testing stage gave results with B = 3 in a reasonable time, and we did not have time to experiment with various beam sizes for this paper.
3. Adam vs RMSprop optimization
    a. Lopyrev uses RMSprop optimizer. As Adam optimizer is popular as an initial optimizer in most NLP papers/research, we employed the Adam optimizer in a few experiments as well. We acknowledge that we did not tune RMSprop as we ran out of time and financial resources, which may have lent the models in experiments 6 and 7 to not perform as well as they could have.

Evaluation and Error Analysis

Most papers [1] published on abstract summarization have been done on the CNN/Daily Mail dataset and typically report full-length F1 scores of ROUGE-1, ROUGE-2, and ROUGE-L. For our paper, we acknowledge the fact that our usage of WikiHow does not have a comparable baseline for comparison. However, we report our results using ROUGE-1 scores, particularly ROUGE-1-P (precision, or BLEU), ROUGE-1-R (recall), and ROUGE-1-F (F1 score). ROUGE-2 was attempted, but produced extremely low scores, indicating much more potential improvement to be made to our model. A critical limitation of traditional ROUGE based metrics is that it does not consider equivalent paraphrasing, for example, using synonyms of words in reference summary. Manual observation of samples reveals three classes of outcomes: One, where the generated summary is poor; two, where the generated summary is very close to the reference text; and--the most interesting--three, where the generated summary is reasonable given the document but very different from the reference summary. Here are samples of each of the 3 classes described above:

Category 1 - Generated summary is poor

Text	**Reference**	**Generated**
if you are having a hard time penning your speech , consider turning to famous acceptance speeches for ideas on how and how not to proceed . modern history is full of examples of great and terrible acceptance speeches that can be used as inspiration …<more text>... as an oddball example , consider joe pesci's oscar acceptance speech . …<more text>...	look for inspiration from the greats .	give her a nickname .
a hard disk is built to store and access the large amounts of files that you want to preserve in your computer . it stores the songs , files , movies and …<more text>... use the disk cleanup utility to help you keep the disk space clean and safe , and speed up windows xp easily and quickly . windows xp comes with disk cleanup utility that is built …<more text>...	keep your disk safe and clean .	clean up .

Category 2 - Generated summary is close to reference

Text	Reference	Generated
once you receive proof that the petition , summons , and related documents have been served on your spouse , you must file your proof of service form with the court where you filed your petition . …<more text>... keep in mind that if your hearing has not yet been set , you also must have your spouse served with notice of the hearing .	file your proof of service .	file your proof of service .

Category 3 - Reasonable prediction but using different words resulting in a low score

Text	Reference	Generated
your baby will be delighted by your funny faces , and may try to imitate you if she can . making funny sounds , sticking out your tongue , or twisting your face into a funny shape will be entertaining for a baby . babies love to be talked to and have their attention focused on you . you are likely your newborn baby's favorite toy .	make your face entertaining .	play with your newborn .
research has shown that prolonged eye contact can increase feelings of attraction between two people . if you are trying to win her over , try looking into her eyes with a penetrating gaze . …<more text>... if a long gaze does not seem appropriate yet or you are just too nervous to hold a long gaze , try giving her frequent , quick glances .	gaze into her eyes .	make eye contact .

Some of the other observations we had while evaluating our results:

1. **Wikihowsep titles aren't perfect**
   in several instances the headline does not intuitively derive from the document. For example consider the following:

Text	Reference	Generated
your vet will often recommend antibiotics , particularly if the inflammation is caused by an injury or an underlying bacterial infection cold . antibiotics are also given to help ward off other infections . mycoplasma , bordatella , and chlamydia can all be treated with antibiotics .	give your cat antibiotics .	take antibiotics .

In this example, the document contained no clues that the document is about a cat, and thus the generated summary was unable to provide subject context.

2. **One reference summary per document.**
   A document could have multiple good and reasonable summaries. The wikihow data however, allowed for only a single summary per document.

3. **Pretrained word embeddings.**
   The decision to not train our embeddings, cut our trainable parameters by more than half. For example when training with 4 layer encoder + decoder each with 400 hidden units, there are a total of 156 million parameters, 87 million are related to embeddings thus leaving only 69 million parameters to be trained.

4. **Impact of preprocessing**
   To keep the size of the model manageable, we trimmed the document to 100 words. While this gave us a good boost in performance, it also led to degradation of performance on long documents. For example:

Text	**Reference**	**Generated**
Trimmed portion: you have had access to them all year round . just why do you need to take them camping …<more text>... it is not so much camping as trying to put up with the experience …<continued below>... **Fed to the Model:** ! get wild and leave the electronic gear at home . rediscover conversations , story telling , drawing in the sand soil , and stargazing . none of these activities require complex gear . …<more text>...	consider banning the electronic gadgets .	be creative .

Above, notice how the trimmed portion of the document removed critical pieces how leaving behind electric gadgets could encourage the creative work.

Github links to code used for inference and error analysis

Code for Beam Search:
https://github.com/vivekmids/nlp-summarization/blob/master/beam_search.py
Code for Evaluation (reference from: https://github.com/neural-dialogue-metrics/rouge)
https://github.com/vivekmids/nlp-summarization/blob/master/bleu_rouge_functions.py
Code for Error Analysis:
https://github.com/vivekmids/nlp-summarization/blob/master/Error_Analysis.ipynb

Conclusions

We did not have a strong baseline to compare our results with since none of the previous work reported summarization on the WikiHow dataset. Given the limitations listed in the error analysis, however, we are not too discouraged by our low rouge scores. As almost all of our generated text demonstrated correct grammar and a good portion of them actually captured the essence of the documents, we believe that we are on the right track. Some of the other ideas we would have liked to implement had we had enough time and computing resources:

1. Model tweaks and changes:
    a. Including Bi-directional LSTMs
    b. Trying multiple attention layers, or different variants of the attention layers.
    c. Employ more expressive embeddings, such as ELMO instead of GloVe.
    d. Custom loss function to encourage longer summaries and accept synonyms
    e. Try different optimizers and corresponding hyper parameters.

2. Better metrics for performance evaluation:
    a. Rouge-AR by Sydney Maples at Stanford (https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports/2761938.pdf)
    b. Salesforce proposes a mixed objective that optimizes n-gram overlap with ground-truth summary while encouraging abstraction by combining maximum likelihood estimation with policy gradient (https://www.aclweb.org/anthology/D18-1207.pdf).
    c. Facebook's InferSent generates sentence embeddings (https://github.com/facebookresearch/InferSent) which could be used to calculate the distance between generated and reference sentences as a custom metric.

References

1. Ruder, Sebastian, NLP-progress, (2016), GitHub repository, https://github.com/sebastianruder/NLP-progress
2. K. Lopyrev, "Generating News Headlines with Recurrent Neural Networks," arXiv: 1512:01712 [cs], Dec. 2015.
3. A. See, P. Liu, and C. Manning, "Get To The Point: Summarization with Pointer-Generator Networks," arXiv: 1704.04368 [cs], April 2017.
4. M. Koupaee and W. Wang, "WikiHow: A Large Scale Text Summarization," arXiv: 1810.09305 [cs], Oct. 2018.
5. D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation By Jointly Learning To Align And Translate," arXiv: 1409.0473 [cs], May 2016.

Code References

1. Reference for building encoder-decoder: https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html
2. ROUGE package: https://github.com/neural-dialogue-metrics/rouge
3. Attention layer package: https://github.com/thushv89/attention_keras
4. Sequence to Sequence dimensionality reference: https://medium.com/shortcutnlp/how-to-implement-seq2seq-lstm-model-in-keras-shortcutnlp-f55fadd7f812
5. BLEU metrics: https://machinelearningmastery.com/calculate-bleu-score-for-text-python/
6. Sequence to Sequence pipeline example: https://github.com/NainiShah/News-Headline-Generation