Abstractive Text Summarization with Quasi-Recurrent Neural Networks

Peter Adelson
Department of Computer Science
Stanford University University
padelson@stanford.edu

Sho Arora
Department of Computer Science
Stanford University University
shoarora@stanford.edu

Jeff Hara
Department of Computer Science
Stanford University University
jhara18@stanford.edu

Abstract

We investigate a recent neural net model - Quasi-Recurrent Neural Networks - and their application to abstractive text summarization, specifically generating a headline from the text of a news article. We use an encoder-decoder with attention model. We compare a pure QRNN model as well as a QRNN/RNN hybrid model to a pure RNN model. Particular areas of investigation include speed-up and performance as measured with ROUGE and training/test loss.

1 Introduction

Text summarization - the condensing of input text into a shorter yet still meaningful output - forms a critical natural language processing problem. Text summarization has seen three dominate approaches: extractive, deletion-based compression, and abstractive. Extractive summarization seeks to find the key passages and sentences and use them to construct a summary. Deletion-based compression aims to remove less-meaningful words to shorten a document. The abstractive approach takes a more bottom-up approach, in which aspects of the generated summary may not appear in the original due to paraphrasing or reordering [1]. The abstractive approach is data-driven, and while summary generation becomes more computationally intensive, they offer increased flexibility, which improves the possible summary outputs.

Within text summarization, we focus specifically on generating headlines from news articles, aiming to create short sequences that present a concise summary of the longer article. Such a task offers a quintessential sequence-to-sequence task and is therefore highly appropriate for our experiments.

In this paper, we take an abstractive approach to the task of headline generation, based around a Quasi-Recurrent Neural Network (QRNN) encoder-decoder model[2], a model that combines elements of a Convolutional Neural Network and a Recurrent Neural Network. Our QRNN model also included attention, a concept that has been found useful for many sequence-to-sequence natural language processing problems, including machine translation and text summarization. QRNN's have been found to provide equivalent or superior performance compared to RNN's while providing increased parallelism that offers significantly faster runtime.

For our model, we use pre-trained GloVe word embeddings from the Stanford Natural Language Processing Group, vectors originally trained on Wikipedia and Gigaword datasets[1].

[1] GloVe vectors available here: https://nlp.stanford.edu/projects/glove/

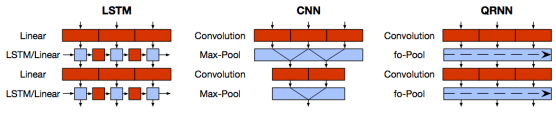
To draw conclusions about QRNN's effectiveness in solving the problem, we developed a baseline RNN model to which we compare the QRNN model. To evaluate the generated headlines, we use the standard ROUGE metric. Section 4 describes the outcomes of the experiments we conducted and Section 5 draws conclusions based on the outcomes.

2 Background

In recent years, within the field of text summarization, abstractive models have gained attention, particularly those using neural-network-based models. Rush, Copra, and Weston demonstrated the effectiveness of a neural-based attention model in abstractive sentence summarization [1]. Various other models, including Long Short-Term Memory models, have proven to be effective at relevant tasks [3]. LSTM and Deep Neural Networks have been shown to be highly effective at sequence-to-sequence semantic translation [4]. However, due to hard dependencies on previous time-steps, RNN's have limits to their parallelization as well as their ability to parse long sequences.

To address these problems, Bradbury, Merity, Xiong, and Socher developed the Quasi-Recurrent Neural Network, which combines convolution and "fo-pooling" (their custom pooling scheme) to build a model that facilitates further parallelization[2]. The convolution can be done in parallel, which is where the QRNN derives its speed-up potential. QRNNs can use either mass convolution or centered convolution. Mass convolution has dependencies only on previous time steps while centered convolution uses information from future time steps. Due to the recent publication of QRNN's in late 2016, there has been a dearth of published papers using the QRNN model. Figure 2 demonstrates the potential of the QRNN in sequence-to-sequence related tasks[2]. QRNN's contain a convolutional layer that feeds into an fo-pool layer. These layers can be repeated. See Figure 1 for an illustration of a QRNN in comparison to other models.

Figure 1: Illustration of a QRNN Model in Comparison to CNN and LSTM [2]



GloVe Vectors offer a powerful tool for embedding word meanings [5]. Using a combination of global matrix factorization and more localized context window methods, GloVe Vectors offer a superior model for word vectors and are used in a variety of contexts as a standard word embedding model[2][6].

Figure 2: Relative Comparison of Speed [2]

Model	Train Time	BLEU (TED.tst2014)
Word-level LSTM w/attn (Ranzato et al., 2016)	–	20.2
Word-level CNN w/attn, input feeding (Wiseman & Rush, 2016)	–	24.0
Our models		
Char-level 4-layer LSTM	4.2 hrs/epoch	16.53
Char-level 4-layer QRNN with $k = 6$	1.0 hrs/epoch	19.41

3 Approach

The problem of abstractive summarization can be thought of as an input containing N words $x_1, x_2, x_3 ... x_n$ where $\forall x_i \in V$ for a vocabulary V and generating an output of M words $y_1, y_2, y_3 ... y_n$ where $\forall y_i \in V$. [2]

A common approach to this framework is to sequentially encode the input and then decode it to generate the output.

[2]Note that due to vocabulary size constraints, not all input words may be part of the vocabulary. They are replaced by an <unk> marker.

3.1 Encoder-Decoder

The encoder-decoder model is based around two neural nets; the first neural net encodes input text into a compressed representation. A word is first transformed into a vectorized representation (GloVe), and that vector is the passed through a multiple-hidden-layer neural network that sequentially combines the text. With a QRNN, this can use either mass or centered convolution. We used centered convolution for greater information usage.

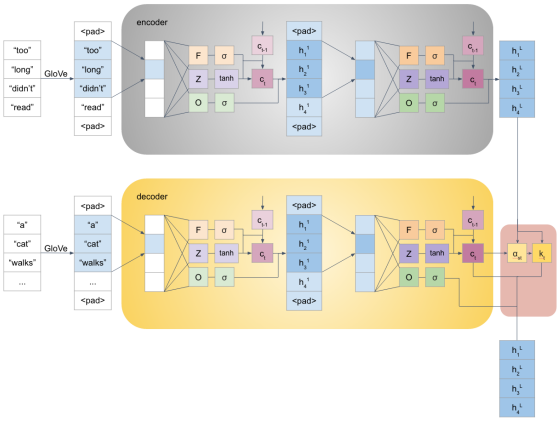
The second neural net accepts the hidden layers generated by the encoder, using neural attention to generate a portion of a headline. This new word is then fed back into the decoder as input, which continues to generate new words until a special end-of-sequence token is output, at which point the headline is complete. During training, the actual headline is fed into the decoder, while during testing, the generated word is used. This is line with the work conducted in [7]. With a QRNN, this must use mass convolution because there can be no dependencies on words not yet generated.

For training, we used the log loss function as follows:

$$\text{-log } p(y_1, ..., y_m | x_1, ..., x_n) = -\sum_{t=1}^{m} logp(y_t | y1..., yt-1, x_1, ..., x_n)$$

All learning rates were constant across models. For Adam Optimization, this was a learning rate of 0.001, β_1= 0.9, β_2 = 0.999, and $\epsilon = 10^{-8}$ and for SGD we used a learning rate of 0.3. Number of layers, hidden units, and batch size were variable across experiments. Furthermore, all RNN's were run with GRU's.

Figure 3: An illustration of the Encoder-Decoder Model, with Attention



3.2 Attention

Both the RNN and QRNN encoder-decoder models used neural attention. Attention is a mechanism that permits a neural net to refer back to previous hidden states for extra information during decoding. Specifically we implement a soft attention mechanism as described in [8]. Selecting the output word is done as a function of the input words, which the attention mechanism weights as learned during training. The weights of attention are passed into a final layer and then used to construct the output. [3]

[3]Usage of soft attention in relation to a QRNN can be found [2]

4 Experiments

4.1 Dataset

Our model was run on 1.5 million data points from the English Gigaword Fifth Edition dataset, made available by the Stanford Linguistics Department. The dataset consists of headline-article pairings from 6 major news conglomerates from a time period ranging 16 years. The headline and article are easily separated.

4.1.1 Preprocessing

For generating headlines, the predictive power of the words beyond the first two sentences has been shown to be dubious [1]. For increased computational efficiency, we included only the first sentence. We also removed all data points that had headlines longer than 25 words or first sentences longer than 50 words. Both headlines and article text were converted to lowercase to ensure consistent comparison.

Another area for a performance optimization was to split the data points into three buckets during training. Looking at our processed data, we made buckets for data points with short, medium-length, and long sentences. This way, we could avoid having excessive padding on smaller data points.

The data was split into training, development, and test sets.

4.1.2 Dataset Issues

The dataset has several irregularities that may have impacted model behavior. First, some headlines are malformed, consisting instead of corrections, publication instructions or other irregular constructions. Examples include

"corrections: for the record"
"sub-graphic-files-nyt".

Some headlines include descriptions unrelated to the article, such as the column the article appears in, as shown with

"net worth column: mutual funds cleaning house"

When the headline is compromised, the article text often is as well, as shown with this pairing:

headline:"sub-graphic-files-nyt"
text: "here is a listing of the new york times news service macintosh graphics files that moved saturday, 01/01/05. included is the corresponding story slug and the size. all graphics have been posted on presslink, newscom and the wieck photo database."

Due to the inconsistent nature of these issues, no effort was taken to combat them across the dataset. Ideally, these malformed headline-article pairs could be filtered or corrected to ensure that the problem remains within the proper domain.

4.2 Tests Conducted

We conducted two separate experiments to compare the QRNN to the RNN in both speed and performance.

Our first experiment was on two separate models, one QRNN-based and one RNN-based. Both neural nets were 4 layers deep with 400 hidden units, ran with batch-sizes of 400, took the first sentence of the text, and were planned to run for 5 epochs. However, this experiment was ended early because there was insufficient time to run it; 4 layers was computationally prohibitive for both models, but we found the QRNN model to be 1/3 faster when compared to the RNN model. With

the resources we had, we decided to run less-computationally expensive models in order to obtain more meaningful results in our given time frame.

Our second experiment was run on three models, one pure QRNN, one pure RNN, and one with a QRNN encoder and an RNN decoder. All three neural nets had 2 layers, 250 hidden units, took the first three sentences, ran for 5 epochs, and had a batch size of 300. We also conducted this experiment using both SGD and Adam Optimizers. Each model took approximately 1.5 days to train for 5 epochs.

4.3 Evaluation Metrics

Two separate evaluation metrics were used to judge the relative performance of models. Training and test loss, as calculated by aforementioned log-loss, formed the first metric while ROUGE formed the other[9]. ROUGE scores were only calculated on headlines generated in the test set.

ROUGE2.0 Java was used, which calculates average recall, precision, and F-score across generated-reference comparisons[10].

5 Results

These experiments led to fairly mixed results. We found that while both the QRNN and RNN models learned and managed to decrease loss with time, only the pure RNN model managed to learn intelligently and output meaningful headlines. Both the QRNN and QRRN/RNN hybrid model failed to do so; they learned to end the sequence, but output headlines that were entirely <unk> markers. While ending the sequence does demonstrate learning, emitting <unk> (the most common character) is a "cheap" way to lower loss and demonstrates what we consider 'artificial unintelligence'.

We believe that running the QRNN and QRNN/RNN hybrid models for additional epochs or with different hyper-parameters may have resulted in more learning, but we cannot guarantee it. The loss/epoch graphs [Figures 4,5,6] for all models demonstrate at least some degree of potential for better learning, although all models also have shallow slopes, demonstrating decreased learning potential.

The RNN was successful in generating coherent headlines that compared well to the ground-truth. A sample of headlines can be found in table 1.

Actual headline	Generated Headline
"dpp chairman touts 'three direct links' with mainland china"	"dpp chairman says china will improve ties with taiwan"
"two palestinians killed in gaza strip clash with israeli army"	"two palestinians killed in gaza strip"

On speed, we found the QRNN to be faster at encoding in comparison to the RNN but slower at decoding. Overall, the QRNN-based models did run faster than the RNN model - a benefit that compounded more with increasing layers and hidden units. However, longer sequence length diminished the QRNN's effectiveness due to the non-parallelizable decoding step, which requires sequential inputs. In fact, having to recompute the convolutions during decoding proves a substantial bottleneck to a QRNN decoder that makes it slower than the RNN counterpart.

Overall, we believe that the QRNN is less effective at abstractive text summarization than it is on other tasks, such as machine translation. Long sequence dependencies and the importance of the decoding step factor into this conclusion. QRNN's also don't capture temporal relationships as strongly as RNN's do, which may be more important in abstractive summarization than machine translation.

Finally, it is clear there are substantial underlying problems with the ROUGE metric; In the case of abstractive summary, which seeks to use words not exactly in the text, common n-grams and subsequences are not the best measure for evaluation, and the results made this clear. Due to the

Figure 4: QRNN Model Loss

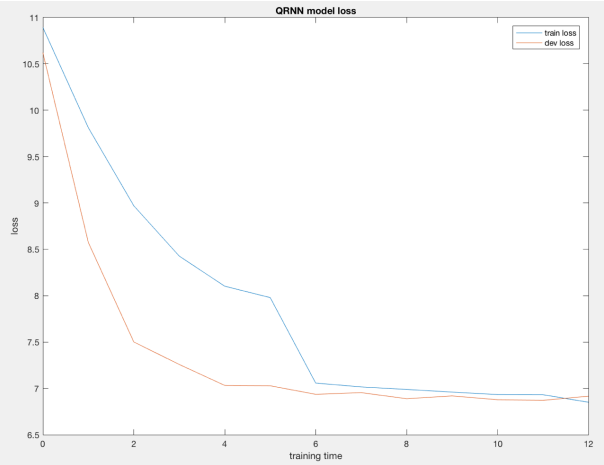
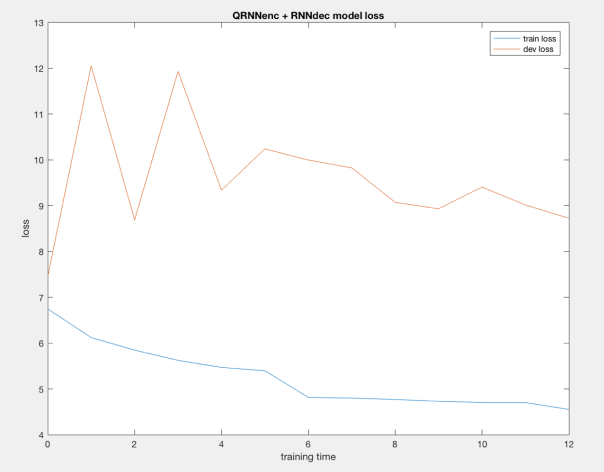


Figure 5: Hybrid Model Loss



results of our QRNN and QRNN/RNN hybrid models, there is no point in reporting ROUGE scores. However, the pure RNN model had a ROUGE-measured average recall, average precision, and average F-Score of 0.18182, 0.22222 and 0.2 respectively.

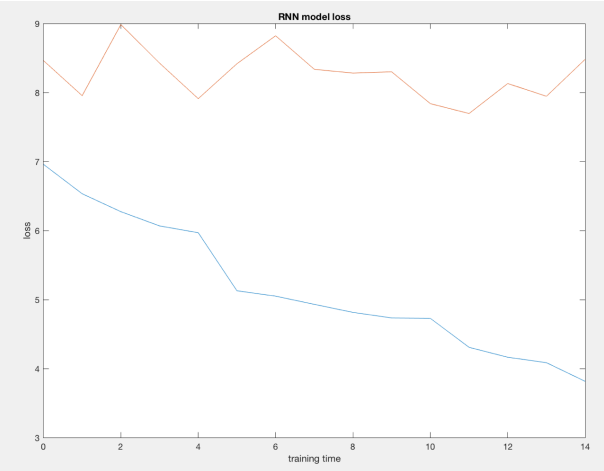
5.1 Future Work

Several extensions to our project have potential to increase efficiency or quality of generated of headlines. First, implementing a beam search in the headline space to ensure the best overall headline is generated offers promise in superior headline generation. Second, pointers to words within the article could be used to implement zero-shot learning and using words outside the vocabulary. Third, the QRNN/RNN hybrid model showed promise and warrants further investigation. Finally, additional layers and the ability to handle longer sequences would make a model that, though it would run more slowly, would also be more powerful, and a user would see even more so the benefits of the QRNN.

Acknowledgments

We'd like to thank Abigail See for her guidance throughout the project. We also thank Richard Socher and James Bradbury availability for answering questions regarding QRNN's. We also would like to thank CS224N and Microsoft for access to the Azure GPU's.

Figure 6: RNN Model Loss



Contributions

Lead areas of work for each member as follows
Jeff: Preprocessing, QRNN Implementation, Poster
Sho: QRNN/Baseline Implementations, Preprocessing, Poster
Peter: Paper, Poster, ROUGE.

References

[1] Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. CoRR, abs/1509.00685, 2015.

[2] James Bradbury, Stephen Merity, Caiming Xiong, Richard Socher. Quasi-Recurrent Neural Networks. arXiv:1611.01576v2, 2016.

[3] Palangi et al. Deep Sentence Embedding Using Long Short-Term Memory Networks: Analysis and Application to Information Retrieval. arXiv:1502.06922, 2015.

[4] Sutskever, I. Vinyals, O. and Le. Q. V. Sequence to sequence learning with neural networks. In Proc. Advances in Neural Information Processing Systems 27 3104?3112 (2014).

[5] Pennington, Jeffrey, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014) 12, 2014.

[6] Kai Sheng Tai, Richard Socher, Christopher D. Manning. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. arXiv:1503.00075, 2015.

[7] Ian Goodfellow, Aaron Courville, and Yoshua Bengio. Deep learning. Book in preparation for MIT Press, 2015.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In ICLR, 2015.

[9] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In Proc. ACL workshop on Text Summarization Branches Out.

[10] http://www.rxnlp.com/rouge-2-0/