NarrativeQA Final Report

John Maxwell, Malavika Srinivas, Sanjana Channabasappa Mallik

https://github.com/CpuKnows/NarrativeQA-Project

Abstract

Our goal was to explore the reading comprehension task with the NarrativeQA dataset (Tomáš Kočiský et al, 2017) and a sequence to sequence model with attention. We wanted to use Tensorflow and Google's cloud computing services to accomplish this since these are popular tools to accomplish tasks in NLP and deep learning, and we had insufficient computational resources without it.

Introduction

We chose the NarrativeQA dataset because it's very recent dataset giving us the opportunity to explore something at the forefront of NLP. We know of no models that have been implement on this dataset besides those used for benchmarking by the authors of NarrativeQA.

There are a few things that separate this dataset from others in reading comprehension and question answering at the moment. The source material for this dataset are books and movies with lengthy context. Some are over 1 million words. In comparison the popular SQuAD dataset (Rajpurkar et al., 2016) uses 1 or 2 paragraphs. For training on NarrativeQA to be tractable preprocessing is required to narrow the context. Another feature that differentiates this dataset is that the answers are not necessarily spans from the context document or multiple choice answers. This requires a deeper semantic understanding of both the context and the question because compared to multiple choice the number of possible answers is large. Compared to multiple choice the number of possible answers is large. In SQuAD the annotators generate the question-answer pairs from the context documents given for training so there's greater similarity in the question and answer phrasing than NarrativeQA where annotators are never shown the source documents. Only summaries from their Wikipedia page. The answers in other datasets are often single words which models like the Attention Sum Reader can capitalize on, but the answers here are lengthier which changes what techniques can be applied.

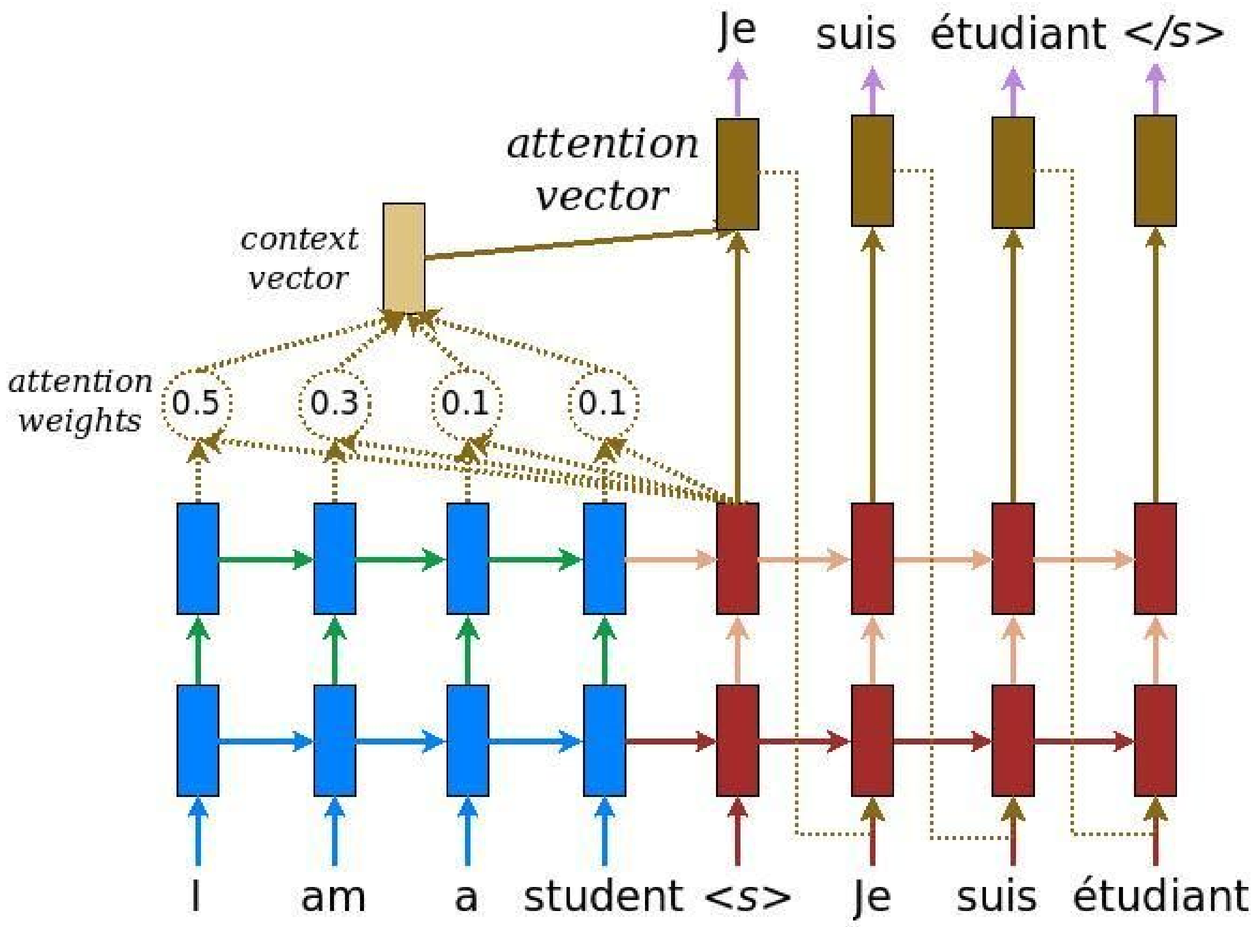
The dataset contains roughly 1,500 documents and 46,000 questions from books and movie scripts. We had issues processing this much information so we decided to narrow it to who questions from books. Movie scripts have different structures from books and the questions were generated from summaries of the movie and not the movie scripts making some questions unanswerable. This gave us about 750 documents and 5,700 questions.

Model

We broke the problem up into 2 main steps following the lead of the NarrativeQA paper. First, we use IR techniques to narrow the context from the entire book to a manageable amount then we applied a model for generating answers from the new smaller context and the question.

For preprocessing we experiment with tf-idf vector similarity and averaged word vector similarity. From our experiments the averaged word vectors worked better and it was simpler to implement in our modeling pipeline. To create the vectors for comparison we moved a sling window of three consecutive sentences over the document choosing the set with the greatest cosine similarity with each relevant question vector. This gave us a context of between 40 and 120 words in most cases which was small enough for us to model.

To generate answers we applied the context-question pairs to a sequence to sequence model using Luong (Luong et al, 2017) style attention. We used this model because we would need to generate mixed length answers where the length could depend on what the question was asking. For example, “Who took the dog for a walk?” is looking for a one or two word named entity whereas “Who is Dave?” is looking for a more descriptive answer. We incorporated an attention mechanism because it helps the model learn which words in the context/question are important signals for a given answer word.



Seq2seq with Luong attention

We briefly describe the seq2seq with Luong attention architect here for completeness. The model is broken up into 2 main sections: the encoder which takes in the context-question sequence and the decided which generates answers. We concatenate the 3 sentence context we chose earlier with the question, separating them with a delimiter <d>. We feed the context in before the question because English is read left to right so most of the information flows that direction as well. In this way the information from the context flows to the question. The encoder will read this input until it reaches a start of sentence token <s>. At this point the encoder feeds its final hidden state to the decoder which starts generating words one at a time until it generates an end of sentence token <s/>. The decoder outputs a vector of logits corresponding to each word in the vocab choosing the maximum value as the final output. At training time we will calculate and back propagate the loss between the correct answer word and the chosen answer word through the network. At inference time we feed the vector of logits at time step t as input into the hidden layer at time step t+1.

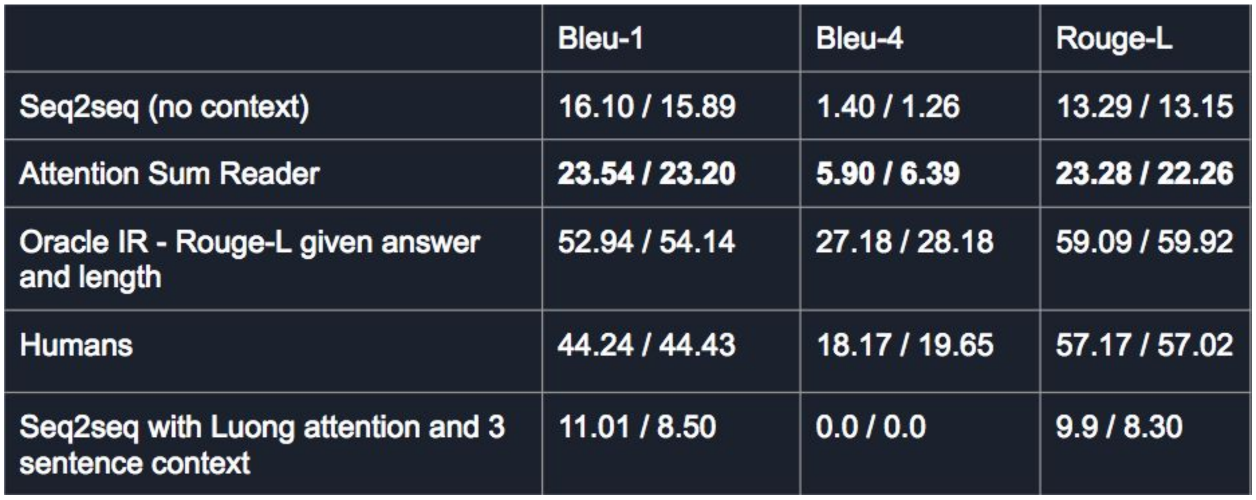
Attention is incorporated between each encoder hidden layer output at the current decoder hidden layer output. A weighted sum of the softmax of each encoder hidden weight is used to create a context vector. The context vector is put through a squashing function, in this case tanh(), with the decoder weights for the final decided weight.

We used a few tricks to better train the seq2seq model. Batching provides a way to speed up training whole reducing variance during an epoch of training, but in order to train in batches the input must all be the same size. Since our context-question pairs are different lengths we need to pad some and truncate others. Given the variance in the lengths this either greatly increase the size of some inputs with padding or greatly reduced the information in others by truncating them. We used the group_by_window () function in Tensorflow to group training samples into similar length buckets which limited the amount of padding and truncating we'd need to do in a particular batch.

Results

We measure the success of our model with Bleu-1, Bleu-4, and Rouge-L in order to compare it with the original paper. These measurements are also appropriate since we will be generating multi-word answers for plenty of questions. Bleu score measures precision by how many words in the model generated answer appear in the annotator answers. Bleu-1 scores on a unigram level and Bleu-4 on a 4-gram level. Rouge-L measures recall by how many words in the annotator answer appear in the model generated answer.

One thing to note about this dataset is that the maximum Bleu/Rouge score of 100 is not the goal. The inter-annotator agreement is the threshold we’re striving for. Given the expressiveness of language there might be multiple correct answers to a question. For example, if two answers are “Queen Elizabeth” and “Queen Elizabeth of England” which one is more correct is a matter of opinion. The results of our experiment are in the table below.



	Bleu-1	Bleu-4	Rouge-L
Seq2seq (no context)	16.10 / 15.89	1.40 / 1.26	13.29 / 13.15
Attention Sum Reader	**23.54 / 23.20**	**5.90 / 6.39**	**23.28 / 22.26**
Oracle IR - Rouge-L given answer and length	52.94 / 54.14	27.18 / 28.18	59.09 / 59.92
Humans	44.24 / 44.43	18.17 / 19.65	57.17 / 57.02
Seq2seq with Luong attention and 3 sentence context	11.01 / 8.50	0.0 / 0.0	9.9 / 8.30

Experiment Results

The Seq2seq (no context) and Attention Sum Reader are baseline models from the paper. Oracle IR is the span from the context documents with the highest Rouge-L score. It is impossible for any span prediction model to achieve a higher score than this. “Humans” is the inter-annotator agreement and the last entry is our model.

Analysis

We would expect our results to be better than the seq2seq model given no context (only the question-answer pairs). We fell short because we ran into a number of issues getting our tools to work together for training which cut short the amount of time for analyzing our results and making adjustments. This included running into memory limits with the 12Gb K80 GPUs we were using. We expect that the authors used GPU clusters because their best performing models used contexts of up to 4,000 words.

There could be a number of issues holding our model back. We may have an information bottleneck requiring us to increase the width of the hidden layers. Better methods of narrowing the context might be necessary as the model can only generate answers based on what it’s seen. If we’re giving it the wrong context this could make it harder for the model to learn by adding misinformation. We could estimate this by creating an Oracle IR model for only our 3 sentence context as the authors did for the whole documents.

A success in our project was coming away with a much better understanding of Tensorflow and how to utilize and manage a project with cloud computing resources. None of us had experience with these coming into this project and these lessons will be very valuable in future projects.

Future Experiments

This is an interesting problem and there are many ways we could expand on it in future work. As we mentioned above the context we retrieve from the document has a large effect on the model’s performance so improving the IR preprocessing would be a good first step. This dataset was created in a way to limit the effectiveness of span prediction by using different material to generate question-answer pairs. Even so the upper threshold on span prediction is quite high. We’d like to experiment with span prediction methods (Yu et al., 2018) since these have had a lot of success in other QA datasets.

A way to capitalize on the dataset domain is the construct representations of the characters and narrative structure as explored in (Iyyer et al, 2016). The authors construct use unsupervised techniques to create a representation of the relationship between 2 characters over the course of a story. We limited our dataset to who questions and the answer to the majority of who questions were named entities. We believe performing coreference resolution on named entities in the context would improve these answers. Performing coreference resolution on the entire document would take way too much time. Another way to represent the documents would be constructing a knowledge graph.

References

Mohit Iyyer, Anupam Guha, Snigdha Chaturvedi, Jordan Boyd-Graber, and Hal Daum´e III. 2016. Feuding families and former friends: Unsupervised learning for dynamic fictional relationships. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1534–1544. Association for Computational Linguistics.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, Edward Grefenstette. 2017. The NarrativeQA Reading Comprehension Challenge.

Minh-Thang Luong, Eugene Brevdo, Rui Zhao. 2017. luong17. In *https://github.com/tensorflow/nmt*

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, Quoc V. Le. 2018. QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension. In *ICLR*.

Hyperparameters

Best model

Architecture: Seq2seq
Attention: Luong attention
Hidden layers: 2
Hidden units: 128
Dropout: 0.2
Embeddings: trained with model

Max encoder sequence length: 100
Max decoder sequence length: 50
Optimizer: ADAM
Learning rate: 0.01