Reading Comprehension for SparkNotes Question Answering

Alexander Hathaway
ahathaway@berkeley.edu

Abstract

Reading comprehension for texts with lengths longer than a few paragraphs, such as novels, plays, and movie scripts, presents a difficult challenge. Over the course of tens and hundreds of pages, a myriad of plot events and character interactions creates a rich text to draw insights and conclusions from. Typically, evaluating reading comprehension can be accomplished with question answering tasks. In an attempt to measure reading comprehension levels against a typical middle or high school student, we draw upon multiple choice quiz questions from the popular literature study guide website SparkNotes. We ask young students to consume classic texts by the likes of Shakespeare and Dickens, and so we can attempt to teach machines how to consume these same texts in an effort to emulate human understanding. Drawing from quiz questions available on SparkNotes, I present a new multiple choice dataset to be used as an evaluation technique for reading comprehension on longer narrative texts. I also introduce a transformer pointer generator model for abstractive question answering on this task.

1 Introduction

Reading comprehension for larger documents requires a combination of several natural language processing tasks - retrieval, question answering, and answer evaluations. Current models are unable to hold an entire book with enough detail to answer specific questions about specific passages and often larger scale themes. To solve for this, often a question is used as a search query to narrow down the entire text into relevant subsections that are more applicable. Books on SparkNotes cover classic literature including texts like The Iliad, A Tale of Two Cities, and Frankenstein.

2 Background - Literature Review

There are several data challenges that align similarly with the task of question answering across longer documents for SparkNotes quizzes. One of the earlier challenges, MCTest (Richardson et al., 2013), is a collection of short stories with multiple choice answers. While there are some similarities with a SparkNotes quiz, the stories are much shorter, and with only 660 stories and 2640 questions the dataset is used more as an evaluation method rather than potential training data.

SQuAD (Rajpurkar et al., 2016) takes a different approach and instead asks a set of questions for a given piece of context test with the goal of identifying answers within the text verbatim. With the introduction of ELMo and BERT, the community has been able to obtain incredible results for this extractive question answering task. The current F1 scores currently outrank human performance. Unfortunately, many of the questions asked in a SparkNotes quiz do not have answers that reside directly in text which puts an upper limit on the ability for extractive question answering to work.

MS MARCO (Nguyen et al., 2016) gets slightly closer to matching the problem, as it presents questions along with several snippets of text. The goal is to identify the answer amongst these multiple passages, and can generally be done in both an abstractive or extractive manner successfully.

Children's Book Test, Book Test, NewsQA, SearchQA also present similar challenges that include some combination of passage retrieval and question answering.

Most similar of all, however, is the NarrativeQA Reading Comprehension Challenge (Kocisky et al., 2018). The authors collected a combination of novels, plays, and movie scripts largely from

Project Gutenberg, and then paired them with user generated question answer pairs. The goal was to ask questions that couldn't be answered with simple text lookups, but rather questions that required comprehension of the underlying narrative. The user generated question answer pairs were generated with a summary of the text as context, meaning that the questions and answers often entailed larger thematic and plot points instead of direct passage questions. In this sense the goal of the passage matches with the SparkNotes challenge incredibly well.

Within the NarrativeQA challenge, there are two sub problems. The first is to answer questions using the summary as context. The second is to use the entire text as the only context. The SparkNotes challenge relates with the second of these sub problems, and unfortunately the only reference to the full text sub problem lies within the original NarrativeQA paper. All other explorations of this dataset have focused on the summary problem which more closely aligns with the datasets mentioned above. These challenges are evaluated on a combination of BLEU-1, BLEU-4, and ROUGE-L scores.

The original NarrativeQA paper uses an implementation of Attention Sum Reader (Kadlec et al., 2016) to answer questions. This approach relies on the answer being contained within the provided passage(s) and achieves a BLEU-1 score of 20.0. Building on this extractive approach, an implementation of the Bidirectional Attention Flow for Machine Comprehension serves as the baseline for purely extractive methods with a BLEU-1 score of 33.45 on the summary challenge.

The state-of-the-art submission for this problem comes from MASQUE (Nishida et al., 2019, which also holds the current high score for the MS MARCO challenge. This approach moves away from the more common extractive techniques, and instead uses abstractive summarization techniques to generate text that may or may not come from the given context. Relying primarily on a transformer model combined with passage ranking, and a pointer-generator (See et al., 2017) decoder, the model obtains a BLEU-1 score of 54.11 on the summary task.

3 Methods - Design and Implementation

To obtain a collection of suitable quizzes from SparkNotes, I had to collect both the quiz questions and the source text. To do this, I cross checked SparkNotes titles that were available for free with Project Gutenberg, and were also included on the list of texts for the NarrativeQA challenge with the hopes of combining the existing questions from that challenge as additional training data. After filtering, this produced 53 unique texts. I then created a web scraper to comb through SparkNotes quizzes available on their website for each of the texts which resulted in 4533 multiple choice question answer pairs.

This number of question answer pairs is certainly not enough to train a model on, and so I used the dataset made available with the NarrativeQA challenge which added an additional 46,765 question answer pairs across 1,572 unique texts.

3.1 Data Collection

To obtain a collection of suitable quizzes from SparkNotes, I had to collect both the quiz questions and the source text. To do this, I cross checked SparkNotes titles that were available for free with Project Gutenberg, and were also included on the list of texts for the NarrativeQA challenge with the hopes of combining the existing questions from that challenge as additional training data. After filtering, this produced 53 unique texts. I then created a web scraper to comb through SparkNotes quizzes available on their website for each of the texts which resulted in 4533 multiple choice question answer pairs.

This number of question answer pairs is certainly not enough to train a model on, and so I used the dataset made available with the NarrativeQA challenge which added an additional 46,765 question answer pairs across 1,572 unique texts.

3.2 Information Retrieval

The next step was to build an information retrieval method that could find pertinent passages for a given hand at a length that was manageable for current question answering models. Queries often pertain to events that happen across multiple instances and characters who evolve throughout the text, and so ideally a semantic approach to capture this nuance would be best. While semantic approaches using Neural IR methods, such as the methods described using BERT (Dai et al., 2019), show promise, I used a simple tf-idf information retrieval method for its combination of simplicity and efficacy.

To break up texts into smaller, separate documents to search on, I evaluated fixed character width, full sentence combinations, and paragraph splits. Paragraphs often serve as a structural way to self-contain complete ideas, but identifying paragraph splits across different text structures like plays and scripts, along with different types of encodings proved difficult, at which point I split the document by sentences, and combined sentences to maximize the number of sentences in a document within a character limit. Ultimately the task of information retrieval serves as a key limiting factor in the question answering task, as without any semblance of an answer embedded in the returned documents, the model is presented with too much noise to answer questions effectively.

3.3 Proposed Transformer Model

The model most closely follows the structure of an abstractive summarization task, except instead of only taking a passage context and summary, there is also a question to pair with the passage. The model architecture looks as follows:

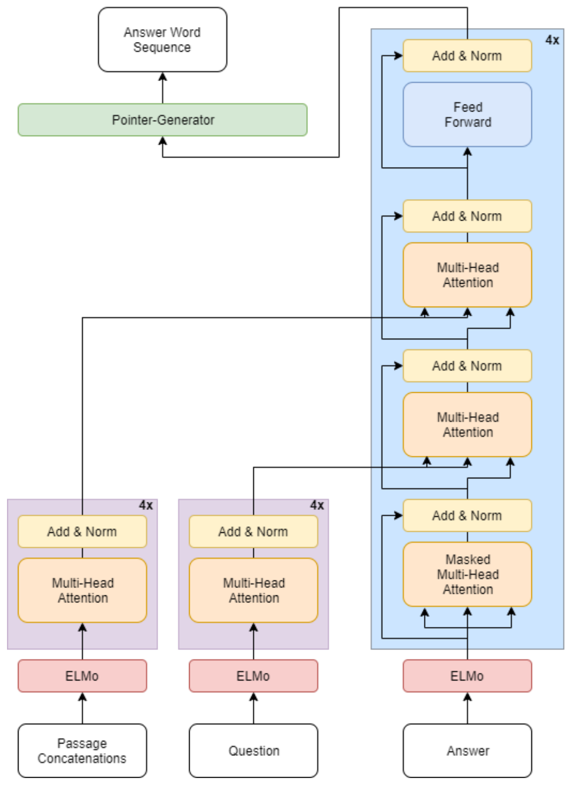

Figure 1: Model Architecture

3.3.1 Word Embeddings

For each of the passages, questions, and answers, the first step was to send the tokens of each through a shared embedding layer. I first experimented with using a fixed GloVe (Pennington et al., 2014) embedding layer combined with positional encodings, which provided substantial improvement over an embedding layer without predefined weights. To better capture contextual representations, I fed the texts into an ELMo (Peters et al., 2018) embedding layer which proved more effective than the GloVe layer.

3.3.2 Encoder

The question and passage are sent through an encoder with a set of transformer encoder blocks for each. The block consists of two sub-layers: a self-attention layer and a position-wise feed-forward network. For the self-attention layer, I used a multi-head attention mechanism (Vaswani et al., 2017) with 4 heads. The feed-forward network consists of two dense layers, with the first utilizing a relu activation function. After both the multi-head attention step and the feed-foward layer, I use a normalization layer and a residual connection to the original input. As the inputs are fed through a padded batch data generator, I use padding masks to ensure any padding is considered for attention.

This process is repeated as 4 stacked encoder layers, and generates an output for each token in the sequence with the dimensions [batch size, input length, dmodel] where input length corresponds to the max batch length for either the passage or question respectively and dmodel refers to the embedding dimensions produced by ELMo which is 1024.

3.3.3 Decoder

Similar to the encoder layers, the target set of tokens is first sent through a transformer decoder block consisting of self-attention and a feed-forward network. Next, to capture context from the question and passage, I feed the outputs of the encoder into subsequent layers of the decoder. First, the outputs from the decoder transformer block are fed into the next transformer block as the query input to the multi-head attention, while the encoder outputs from the question are fed in as the key and value before going through a feed-forward network and a normalization layer.

The outputs of this block are again fed into another block as the query for the multi-head attention step while the outputs of the encoder for the passage are fed in as the key and value input. This three step combination is repeated n times before being fed into a pointer-generator (See et al., 2017). While the abstractive summarization route was chosen partially because answer tokens are not always included as exact spans in the text, often parts of the answer are included in the passage, and so I feed the results of the decoder along with the attention distribution from the passage to combine with the vocabulary distribution in an effort to give more weight to tokens that held significant attention from the passage when generating new tokens.

3.4 Evaluation

Once the decoder generates answer tokens for each question, I then compare the generated text with the available multiple choice answers. I run the generated output and all available answers through an elmo embedding layer, this time capturing an embedding for the entire sequence instead of individual tokens. The answer option with the highest cosine similarity to the generated answer is chosen as the predicted answer.

4 Results and Discussion

SparkNotes Quiz Results				
Model	QAE	QAB	BLEU-1	BLEU-4
BiDAF	25.55%	29.33%	.2486	.2927
Transformer	30%	30%	.2	.2
seq2seq LSTM	24.87%	25.02%	.2	.2

5 Conclusion

As noted above, there are a number of question answering challenges, and results when the answer can be determined from a given passage are quite good. The place of focus that would likely improve results the most would be on the information retrieval side to have higher confidence that returned documents contain the necessary information to answer a given question.

References

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ.

American Psychological Association. 1983. *Publications Manual.* American Psychological Association, Washington, DC.

Association for Computing Machinery. 1983. *Computing Reviews*, 24(11):503–512.

Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. 1981. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133.

Dan Gusfield. 1997. *Algorithms on Strings, Trees and Sequences.* Cambridge University Press, Cambridge, UK.