CS546 Project Report
Narrative Question Answering on Summaries

Prashant Jayannavar(paj3), Dhruv Agarwal(dhruva2), Rishika Agarwal(rishika2)

August 2, 2018

Abstract

We explore the task of question answering for reading comprehension on the very recent NarrativeQA dataset proposed in Kočiskỳ et al. (2017) – also called the NarrativeQA Reading Comprehension Challenge. The challenge proposes two RC tasks – one on full stories (books and movie scripts) and the other on abstractive summaries of those stories. We make an attempt on the latter. The dataset poses a significant challenge compared to previous RC datasets as the answers in the corpus require an ability to integrate information from various parts of a passage, construct higher level abstract representations, reason over them and then produce an answer in one's own words. This thus intends to push the field to go beyond the more simpler answer span prediction methods for example. We, in this project, build a model for the summaries task. The model is an extension of the BiDAF model originally developed for span-based QA on the SQuAD dataset and used as the best baseline model in the original NarrativeQA work.

1 Introduction

QA tasks for RC can be formulated in different ways with varying levels of difficulty. Some tasks provide a passage of text and pose questions in the form of multiple choice questions (MCQs), where the system has to "select" the correct option from the candidate choices. Thus, it can be seen as a classification or prediction task. Yet another formulation is the case where, given a passage, as well as a huge corpus of text (curated from Wikipedia or other such source), the system is posed with some questions. Answering these requires common sense knowledge in addition to the information given in the passage. Thus, the system has to "generate" an answer, taking into account the given text while also building upon some sort of common sense knowledge extracted from the huge text corpus provided. These can be seen as two extreme ends of QA in terms of hardness. Span-based QA lies between these two extremes, as the answer is in the form of a span of the summary text. In this project, we study the summaries task of NarrativeQA Reading Comprehension Challenge proposed in Kočiskỳ et al. (2017), which requires generating an answer to a question posed on a short summary text of a story. It does not require any common-sense knowledge.

Most RC datasets and tasks are dominated by questions that can be solved by selecting answers using superficial information; they thus fail to test for the essential integrative aspect of RC. This aspect requires integrating information and reasoning about events, entities, and their relations across a full document. It requires creating questions that examine high-level abstractions instead of just facts occurring in one sentence at a time. Such questions require the formation of more abstract representations about the events and relations expressed in the course of the document. This was the primary motivation behind the creation of the Narrative QA dataset and task. The dataset consists of stories, which are books and movie scripts, with human written questions and answers based solely on human-generated abstractive summaries. Correspondingly there are two tasks proposed: QA based on the full stories or QA based on the summaries alone. We

make an attempt on the latter.

The BiDAF model originally proposed in Seo et al. (2016) is a popular one for span based QA and is also used as the best baseline model in the original NarrativeQA work. We extend BiDAF for the NarrativeQA task. We adopted the overall structure of this model and modify the output layer, loss function and evaluation scheme for our task.

2 Previous Work

Kočiskỳ et al. (2017) propose three neural baselines on the summaries task. The best performing is the BiDAF model which is designed to predict answer spans. It works well on the simpler SQuAD task proposed in Rajpurkar et al. (2016) as the answers inherently are meant to be spans in the passage. However it achieves a Bleu-4 score of only 15.53 on the summaries task in NarrativeQA (more scores like Bleu-1, METEOR, etc. can be found in Kočiskỳ et al. (2017)). Compared to human performance there is still a lot of room for improvement. The model also does answer span prediction and inherently the answers are not meant to be spans in the summary or the full story. There is thus a need to go beyond simple span prediction.

Given that this a very recent dataset and task and a challenging one, not a lot of prior work exists on this task specifically. One such work is by Tan et al. (2017). They attempt to solve the stories task in two steps. First, use a retrieval model to select passages from the story that are relevant to the question. Second, given the relevant passages, encode them suitably and use a decoder to generate the answer.

3 Dataset

The model was trained and tested on the NarrativeQA dataset, which is a new reading comprehension dataset, consisting of questions based on books from Project Gutenberg or movie scripts scraped from the web. There are in total 1567 stories, which though less in number compared to other datasets but the documents are large enough to provide lexical coverage and diversity. The questions and answers are natural human-generated, and answering questions requires referencing large parts of the comprehension, rather than short local spans.

4 Model Description

The model used for this task is based on the Bidirectional Attention Flow (BiDAF) model for Machine Comprehension which achieved state-of-the-art results on the SQuAD dataset when it was published. It is a hierarchical multi-stage model which consists of six layers. Fig.**?** shows the architecture of the original BiDAF model for span-based answer prediction. Our model is an extension of this and retains all layers except the output layer. These are as follows:

- **Character embedding layer** - it maps each word to a vector space using character level CNNs
- **Word embedding layer** - it maps each word to a vector space using pre-trained word embedding models like Word2vec or Glove.
- **Contextual embedding layer** - it gives separate refined embeddings for the context (summary) and query, by passing the word and character embeddings through bi-directional LSTMs. This captures

temporal relationships between the query words, and the summary words (separately). The context embeddings are denoted by $\mathbf{H} \in R^{2d \times T}$ and the query embeddings by $\mathbf{U} \in R^{2d \times J}$

- **Attention flow layer** - it takes the contextual embeddings of the summary and the query, and uses attention to link the two. Two attentions are computed : query to context, and context to query, to identify the relative importance of each term in the context for a word in the query, and vice-versa. A shared similarity matrix $S \in R^{T \times J}$ is used for modeling these attentions.
  $S_{tj} = \alpha(H_{:t}, U_{:j})$ denotes the similarity between the t^{th} context word and the j^{th} query word. α is a trainable scalar function for computing the similarity between two vectors. The model uses $\alpha(h,u) = w_{(s)}^T[h;u;h \circ u]$ where $w_{(s)}^T$ is a trainable weight vector. This shared similarity matrix is used to compute the two attentions as described below :

  - **Context-to-Query Attention** : For every context word x_t, an attention vector is computed over all the query embeddings $q_j, j = 1,2,..,J$, which tells the relative importance of each query word for the given context word. The attention vector is given by $a_t = softmax(S_t) \in R^J$. Multiplying the attention vector by the query word embedding matrix U gives the attended query vector $\tilde{U}_{:t}$ for the context word. And the concatenation of the attended vectors for all the context words gives the attended query embedding matrix $\tilde{U}$.

  - **Query-to-Context Attention** : For the entire query, an attention vector is computed over the context embeddings q_t, which tells the relative importance of each context word for the given query. Note that there is just one common attention vector for the entire query sentence, unlike C2Q, where we had separate attention vectors for each context word. The attention vector is given by $b = softmax(max_{col}(S)) \in R^T$ Multiplying the attention vector by the context word embedding matrix H gives the attended context vector $\tilde{H}_{:t}$ for the query. Since this attended vector is common for the entire query, it is tiled T times to give the attended context embedding matrix $\tilde{H}$.

  The attended context and query embeddings ($\tilde{H}$, $\tilde{U}$), and the context embedding (H) from the previous layer are combined to give the output of the attention flow layer(G).
  $g(h,\tilde{u},\tilde{h}) = [h;\tilde{u};h \circ \tilde{u};h \circ \tilde{h}] \in R^{8d \times T}$, where T is the number of words in the summary or context, and J is the number of words in the query.

- **Output Layer** - This is where our model deviates from the BiDAF model. The original model predicted the start and end indices of the span for the answer in the context/summary. However, our task requires generating free-form answers which are not restricted to simple spans of the summaries. We thus use an encoder-decoder framework for the output layer that is capable of generating such answers while leveraging the highly contextualized and attended word embeddings from the previous BiDAF layers. The input to the encoder BiLSTM is the output of the modeling layer shown in 1. The last hidden states of the forward and backward LSTMs are then concatenated and used to initialize the initial hidden state of the decoder LSTM. The output of the decoder is then the predicted answer. We use greedy decoding.

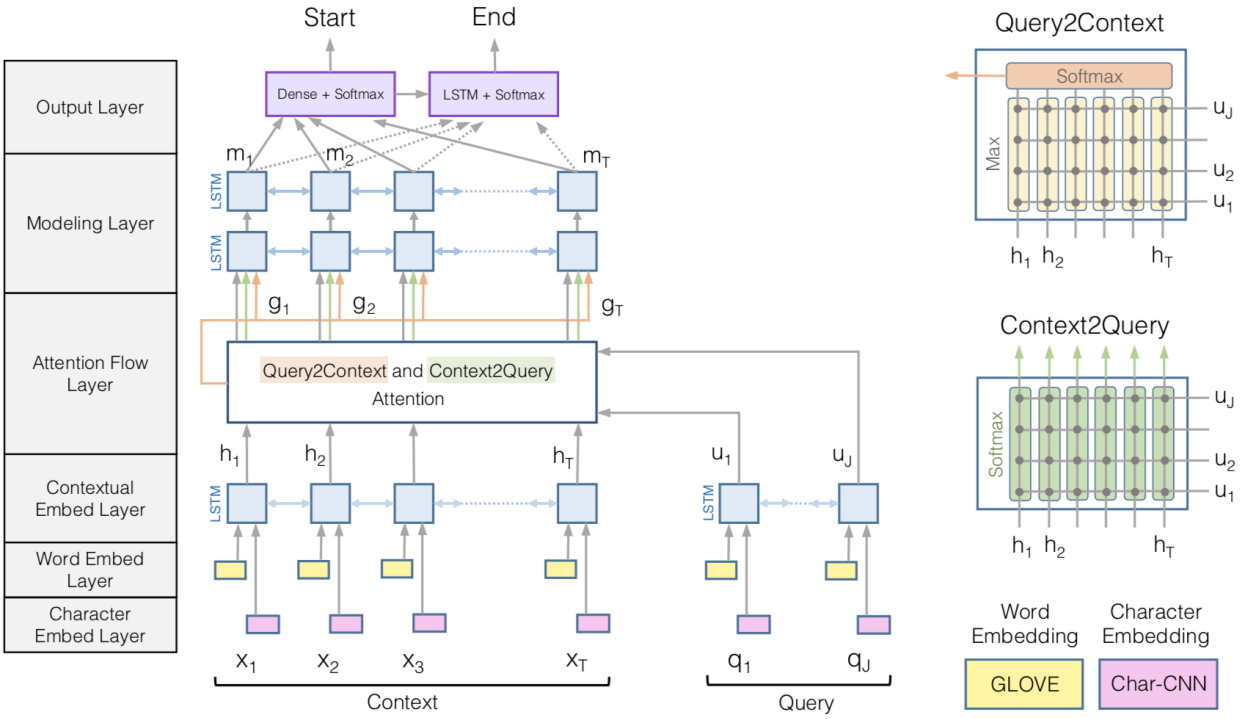

Figure 1: The original BiDAF model from **?**

5 Training

Cross-entropy loss between the predicted probability distribution and the reference answer's distribution over the vocabulary is used to train the model. The probability distributions over the vocabulary are vectors of length V. The predicted distribution is continuous, while the reference distribution is binary, ie, the i^{th} element is 1 if the i^{th} vocabulary word occurs in the answer, and 0 otherwise. The Adam optimizer was used to minimize the loss.

5.1 Evaluation

The generated answers were evaluated against the reference answers on n-gram Precisions and Bleu score. We set n = 4, so four precision values were calculated. The Bleu-4 scores are given in Table 1.

6 Implementation Details

Our work involved the following main steps to extend `https://github.com/allenai/bi-att-flow` which is implemented to work only for the SQuAD task off-the-shelf:

1. Removing all of the SQuAD specific data preprocessing steps
2. Pre-processing the NarrativeQA dataset to feed into BiDAF in the right format
3. Removing the output layer and loss function and replacing them with our own
4. Removing the evaluation mechanism designed specifically for the SQuAD task

5. Interfacing the new output layer with our bleu score evaluation component

All of our code is available at `https://github.com/dhruv100691/CS-546--Narrative-QA` (on the feature/new-output-layer branch but soon to be moved to master).

7 Training details

Some training details are :

- Approximately 4.9 million parameters
- 30 hours training time for 1/4th the dataset, 8500 iterations, and 8 epochs.
- Input data restricted to summaries with length below the threshold of 30 sentences, each sentence not exceeding 30 words. This reduced the dataset to 12,000 summaries, from 32,000 in the original dataset.
- Initial learning rate: 0.001
- Batch size: 10 (we could not train on a bigger batch size because of out of memory error on BlueWaters)
- Character embeddings removed to fit in the GPU memory
- 50 dimensional Glove embeddings for word embedding.

8 Experiments and Results

We ran two main experiments, one with 8000 training samples, and the other with 12000 training samples (we also ran a few other ones they either did not yield significantly different results or ran into problems on BlueWaters).The latter ran for lesser number of epochs, but gave better BLEU-4 score, possibly because of greater vocabulary size due to the greater training data. The results in Table 1 below are shown for the test set.
The original BiDAF model predicting the answer in the form of a span, achieved a BLEU-4 score of 15.53 on the datatset. We hoped to achieve better results than this, but there might be several explanations as to why our model did not perform well. Firstly, we could not train on the whole dataset. This also limited the vocabulary size. Secondly, there is one inherent problem with generative QA : the output is a prediction over the vocabulary of words seen in the training set. Thus, for the test set, which would have many words not there in the training set, especially named entity words, the model will either predict a wrong word from the training set, or the 'UNK' token. Thus, there is need for a way to handle out of vocabulary words rather than simply predicting it as 'UNK'. Span-based predictions do not run into these problems because they do not have to predict a distribution over the vocabulary, but over the indices of the summary. Also BLEU in itself checks for overlap between the generated answer and the reference answer. A more comprehensive metric would be more suitable for generative QA to factor for the various answer variations possible.

9 Problems Encountered

Hardware limitations: We required 12 GB of GPU memory to train the full model, but we had only 6 GB of memory on Blue Waters. Further, we could only run tasks for maximum of 30 hours, whereas our model

#Training summaries	#epochs	training time	Bleu-4
8000	10	17.78 hrs	7.546
12000	7	17.5 hrs	8.173

Table 1: Table of Experimental Results

required more time to train fully. A few of our experiments also ran into these problems while running and crashed, thus rendering them useless.

10 Conclusion and Future Work

We developed a model for generative QA on the NarrativeQA task by adapting the output layer, loss function, and evaluation scheme of the BiDAF model originally designed for span-based answer prediction. Due to hardware constraints, we could neither train the model on all of the data nor for enough time on the reduced data. This yielded relatively poor results compared to the BiDAF baseline used in Kočiskỳ et al. (2017). However, we also recognized some conceptual problems of handling out of vocabulary words for generative QA, and the unsuitability of overlap-based metrics like BLEU scores for evaluating the results. In fact, even the cross entropy loss function which is used to train the model is not suitable for generative QA, because it does not care about the coherence of the generated text, but simply encourages more word-level overlap with the reference answer. A loss function which looks at the entire string of generated text and evaluates a loss which penalizes poorly structured and ungrammatical sentences is needed. We came across Wiseman and Rush (2016) which addressed this problem . So, for future work, we would like to develop a way to handle out of vocabulary words better than just predicting UNK, and incorporate the sentence-level loss function in our model.

References

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The narrativeqa reading comprehension challenge. *arXiv preprint arXiv:1712.07040*, 2017.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

Chuanqi Tan, Furu Wei, Nan Yang, Weifeng Lv, and Ming Zhou. S-net: From answer extraction to answer generation for machine reading comprehension. *arXiv preprint arXiv:1706.04815*, 2017.

Sam Wiseman and Alexander M Rush. Sequence-to-sequence learning as beam-search optimization. *arXiv preprint arXiv:1606.02960*, 2016.