Answering Questions with Neural Models

Ajinkya Zadbuke
CICS
University of Massachusetts, Amherst
azadbuke@cs.umass.edu

Reetika Roy
CICS
University of Massachusetts, Amherst
reetikaroy@cs.umass.edu

Abstract

Question Answering is a task that lies at the intersection of Natural Language Processing and Information Retrieval. We propose and investigate a novel neural architecture combining Dynamic Coattention and Pointer Networks to answer questions from the Stanford Question Answering Dataset (SQuAD). We have performed experiments to see if our model is capable of predicting answers correctly and evaluated this based on the standard F1 and Exact Match metrics.

1. Introduction

Question Answering has long been one of the more difficult challenges in natural language understanding and a focus of research in Machine Comprehension for years. In the simplest scenario in this domain, the question is in the form of multiple choice questions and the model has to pick the most appropriate answer making is a fairly simpler classification task. However in our setup, we intend to concentrate on text-based question answering, where the questions are paired with chunks of text containing the answer. We have used the Stanford QA dataset [1] which has triplets of the form *(Context, Question, Answer)*. Although the answer is present within the context and does not require world knowledge to be inferred, the answer may be of variable length. This is a slightly more complex knowledge base that the machine is expected to understand in order to answer the question.

Over the years many benchmark datasets have been curated for evaluation of this task. Early models that tried to solve this task used sliding windows or parsing or using extensively engineered features extracted from the data to build models. However with the advent of deep learning, building neural models for this task showed significant improvement towards the human performance. This problem in itself is interesting and relevant as being able to infer answers can later be extended to information retrieval, dialogue systems or asking questions to the model about an image or video input to see what the model understands from these beyond the basic task of captioning images or summarizing videos.

We approached the problem by first using GloVe [2] embeddings to get vector representations for both the context and the question and then encoding them using GRU units. We use a two-level attention mechanism to replicate the human behavior of reading both the question and the context simultaneously to leverage the importance of the contents of the context to answer the question. An Answer Pointer Network determines the span of the answer within the context paragraph. In the last section we have presented the evaluation of our model.

2. Related Work

Question Answering has received much attention of late in the deep learning community, with availability of large, high quality datasets and increasingly sophisticated deep learning models. A lot of approaches from architectures built for computer vision tasks have been adapted to NLP problems.

Apart from SQuAD, there are a few other datasets for question answering. BabI, by Weston et al. [3] consists of multiple, clearly defined tasks that target different areas of reading comprehension. The CNN/Daily Mail dataset [4] formulates QA as cloze-form question-answer pairs (answer entities are replaced with placeholders, making the problem analogous to filling in the blanks). For our purposes SQuAD is more than sufficient, although we could test our models over these to gauge generalization performance given slightly different but essentially the same task.

Attention is one of the more significant advances, used in translation by Bahdanau et al. [5] to emphasize parts of the input sentence to concentrate on, conditioned on the preceding words. This was one of the earliest papers that showed how introducing an attention mechanism on a decoder reduces burden on the encoder since the decoder and selectively retrieve parts of sentences where the attention is high allowing information to be spread through the data, relieving the need to encode all of the information in the con-

text. Seo et al. [6], Vaswani et al. [7] and Xiong et al. [8] investigate further on using attention mechanisms to model dependencies without being computationally inefficient.

The Bi-directional Attention Flow (BiDAF) model introduced in [6] explains one of the most popular architectures used for Question Answering and has been used as a foundation for more advanced models. This model used bi-directional attention flow to tackle the issue of early summarization in previously suggested uni-directional attention flow models. It captures complexities between the question and the context through character, word as well as contextual embeddings. In [7] the authors show how attention mechanisms can help avoid the use of convolution and recurrent networks completely in capturing global dependencies between input and output. This mechanism outperforms other traditional ones in terms on the distance over which they can capture these relationships. DCNs [8] were our primary inspiration, and the model proposed uses a dynamic pointing decoder along with a co-attention network which helps estimate the answer spans more correcting by iterating over potential spans and assists in recovering from getting incorrect answers if the model is stuck at a local minima.

Kumar et al. [9] propose incorporating a mechanism for retaining episodic memory through attention and using information from previous iterations as well as inputs. Wang and Jiang [10] extend work done by Vinyals et al. [11], combining Match-LSTMs with Pointer Networks. Match-LSTMs use the concept of premise and hypothesis to match parts of a given context to the question asked, and the Pointer Network is used since the answer is always a part of the input and it uses attention to rescan the input and extract the output from it. We felt this would be an interesting approach to try out, and consequently used pointer nets to predict the start and end spans of our answers.

Yang et al. [12] and Wang et al. [13] explore using combined character- and word-level representations, and self-matching respectively. It is hard to infer if word-level or character-level information will be a better feature for a model or even combinations of both. These papers aggregate both levels of information from a passage using self-matching networks.

3. Approach

3.1. Pre-processing

Since we have pure text data, it is necessary to perform proper pre-processing and structuring of the data for further analysis. We model each context (passage) - question pair as a sequence of tokens. Both are capped at a maximum length of 600 and 100 respectively, based on dataset statistics, and padded at the beginning to account for lengths shorter than the selected size. We remove all punctuation

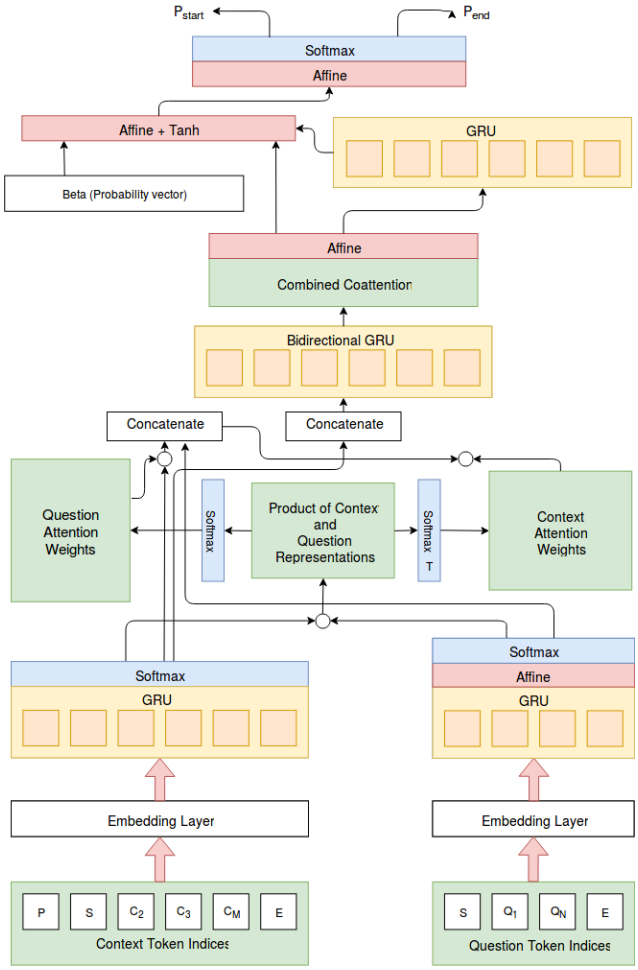

Figure 1. Coattention + Answer Pointer Model

from the sentences as well as trim apostrophe from them so that they can be easily mapped to the embeddings. We do not perform any sort of stopword removal or stemming to preserve sentence structure. These tokens are then mapped to indices of their GLoVe embeddings. We add separate UNK (for Out-of-Vocabulary words) and PAD tokens to the vocabulary, with their embeddings set to zero and negative one. Tokens denoting start and end of the sequence, SOS and EOS respectively, are also included. For the model we are using, it was necessary to find the token spans for the answers within the context to create our target variable. The dataset provided the starting character span for the answer but we decided to get token level spans rather than token level spans to reduce errors at the final classification stage, as selection over 600 tokens would be more sensible to optimize compared to 5000 characters.

3.2. Models

We built three models to tackle this problem. Our initial model consisted of an Embedding layer mapping to-

kens to GloVE vectors followed by a Bidirectional LSTM and Linear layers, of which the latter maximizes probability over start and end character spans. We then focused on including an attention mechanism over the inputs, which gave better results. A more sophisticated model was built later, heavily inspired by DCNs [6], BiDAF [6] and Pointer Networks [10, 11]. Here, we have the same embedding layers as above, separately encoding the context and question input sequences. These are fed to two different unidirectional GRU networks.

$$D = \overrightarrow{GRU}(context) \tag{1}$$

$$Q = \overrightarrow{GRU}(question) \tag{2}$$

The resulting affinity matrix, denoted by L, a product of the outputs from both, is decomposed through softmax layers to attention weights for both context-to-query and query-to-context representations. While this procedure forms the basic attention mechanism, coattention takes it a step further by encoding the context-to-query outputs as weights for the query-to-context outputs, forming a two-level attention representation. This is finally fed through a bidirectional GRU to obtain the coattention encoding.

$$L = D^T Q \tag{3}$$

$$A^Q = softmax(L) \tag{4}$$

$$A^D = softmax(L^T) \tag{5}$$

$$C^Q = DA^Q \tag{6}$$

$$C^D = [Q; C^Q]A^D \tag{7}$$

$$U = [C^D; D] \tag{8}$$

$$C_{coatt} = \overrightarrow{BiGRU}(U) \tag{9}$$

To obtain a prediction for the span, we implement a boundary-based answer pointer network, which takes its input from the previous coattention network. Passing this through a tanh non-linearity and normalizing with log-softmax gives a probability distribution over the context vector indices. To condition the end index on the starting one, the previous output is combined with another GRU and the same affine layers, allowing for temporal relations as well as sharing the same parameters to predict the end index.

$$F_k = tanh(VC_{coatt} + (W^a h^a_{k-1} + b^a) \tag{10}$$

$$\beta_k = softmax(v^T F_k + c) \tag{11}$$

$$h^a_k = \overrightarrow{GRU}(C^T_{coatt}\beta^T_k, h^a_{k-1}) \tag{12}$$

4. Experimental Setup

4.1. Dataset

We are using the Stanford Question Answering Dataset (SQuAD) [1] which is curated from 536 articles from

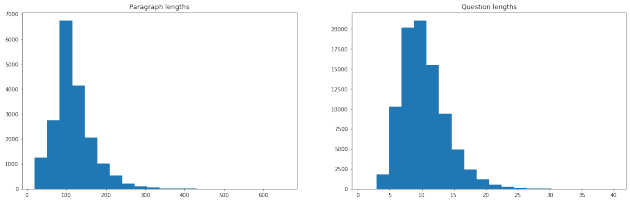

Figure 2. Token-wise lengths for instances in the training data.

Wikipedia over different areas. Each paragraph from the article is considered a passage and has around 5 questions associated with it. This gives a total of 23,215 passages and 107,785 questions. The data was split into a training set (with 87,599 question-answer pairs), a development set (with 30,000+ question-answer pairs) and a hidden test set. We further partition the training data into separate training and validation sets with a 4:1 ratio, and used the provided development set for final testing since we do not have access to the hidden test. Answers are not provided as separate choices, but rather are parts of the same passage, and can be single entities (e.g. Name, Year) or spans of text. The somewhat limits the scope of the task to a more reasonable one of extracting information from only the given content, not requiring any out-of-context knowledge or reasoning.

4.2. Evaluation

Each passage has multiple ground-truth answers (at least 3) associated with it, and the evaluation basis is a macro-averaged F1 score and an Exact string match (EM) metric. Both of these consider token spans from the answer, determining how close the predicted answer is to the actual one. There is an existing evaluation script available, and the dataset paper includes a baseline for human performance as well as a sliding window model. The sliding window model [1] considers the overlap of unigrams and bigrams between the context and the question and tries to derive the answer.

4.3. Experiments

All our work is done in PyTorch, and we implemented the above models from scratch. For training, we use the Adam optimizer, with $\beta_1 = 0.9$ and $\beta_2 = 0.99$, with a learning rate of 0.01. The loss function used is a sum of Negative Log Likelihood of predicted log probabilities of the start and end indices (This is equivalent to Cross Entropy due to applying log-softmax earlier). We decided to use GRUs [15] instead of LSTMs [16] since they are known to give comparable performance and are less computationally expensive. We use 50-dimensional GloVE Embeddings, pre-trained on a corpus of 6B Wikipedia articles (particularly suited to the current dataset). The model is trained for a maximum of 20 epochs, with a batch size of 32 and a hidden layer size of 50. Additionally, we add dropout to the linear and sequential layers ($p = 0.7$).

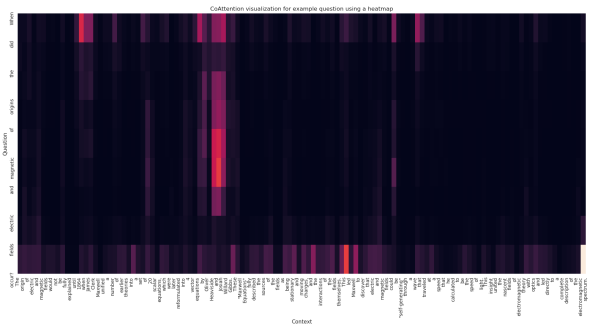

Figure 3. Visualizing the coattention for an example question using heatmaps. The answer of the question is visibly different from other tokens (top-left; Answer=*1864* conditioned on the word *when* in the question.

4.4. Results

Our model performed well given the constraints, achieving an F1 and EM scores of 36% and 18% respectively (Table 1). While not ideal and certainly not close to human performance, we we believe using higher-dimensional representations for the input as well as training for longer times could really help performance. Inspecting the intermediate results, we could see that the model was learning to correctly correlate chunks of text across the context and question. (see Figure 3). Looking at a few incorrect examples, we saw that the model sometimes predicted part of the answer span correctly, having only a few correct words for longer answers. Short answers were more inclined to be predicted exactly, although most of the time the models missed entirely on these.

Models	F1 (%)	EM (%)
Random Guessing Baseline	4.1	1.1
Sliding Window Baseline	20.2	13.2
LSTM model	5.309	2.065
Basic Attention model	21.049	8.732
Coattention + Boundary Answer Pointer	36.115	18.126

Table 1. Results

We looked at some of the predictions made by the model to see how it is performing. Here are some of the prediction results:

Example 1.
Context: The connection between macroscopic nonconservative forces and microscopic conservative forces is described by detailed treatment with statistical mechanics. In macroscopic closed systems, nonconservative forces act to change the internal energies of the system, and are often associated with the transfer of heat. According to the Second law of thermodynamics, nonconservative forces necessarily result in energy transformations within closed systems from ordered to more random conditions as entropy increases.
Question: What is the law of thermodynamics associated with closed system heat exchange?
Possible Answers given in dataset: [Second, Second law of thermodynamics, Second law]
Answer Predicted by model: Second law

Example 2.
Context: The connection between macroscopic nonconservative forces and microscopic conservative forces is described by detailed treatment with statistical mechanics. In macroscopic closed systems, nonconservative forces act to change the internal energies of the system, and are often associated with the transfer of heat. According to the Second law of thermodynamics, nonconservative forces necessarily result in energy transformations within closed systems from ordered to more random conditions as entropy increases.
Question: What changes macroscopic closed system energies?
Possible Answers given in dataset: [nonconservative forces, internal energies of the system, nonconservative forces, nonconservative forces]
Answer Predicted by model: macroscopic nonconservative forces and microscopic

Example 3.
Context: A conservative force that acts on a closed system has an associated mechanical work that allows energy to convert only between kinetic or potential forms. This means that for a closed system, the net mechanical energy is conserved whenever a conservative force acts on the system. The force, therefore, is related directly to the difference in potential energy between two different locations in space, and can be considered to be an artifact of the potential field in the same way that the direction and amount of a flow of water can be considered to be an artifact of the contour map of the elevation of an area.
Question: What is the only form kinetic energy can change into?
Possible Answers given in dataset: [potential]
Answer Predicted by model: kinetic or

The dataset often can have multiple answers of different span lengths and all the possible answers are listed. While evaluation the script checks if the predicted answer matches any of the mentioned answers- if it matches exactly then it is counted in the Exact Match metric and the F1 score checks how much of the actual answer is present in the predicted answer.

In the given examples we see that Example 1 has predicted the exact answer, while in Example 2 the answer is contained in the span but does not match exactly with the

possible answers. In Example 3, however the model has completely failed to predict the answer.

5. Conclusion and Future Directions

Working on this topic was challenging, and we learned a lot about using neural models on real-world datasets. A significant issue we faced and overcame was faulty preprocessing on our part, further emphasizing how important getting the data into the right format is to performance of the model. Using complex, many-layered architectures forced us to delve into the internals of the framework to figure out why a particular unit/layer was behaving the way it did, and helped us develop a deeper understanding of the learning process. For future work, we would like to explore additional attention-based approaches, as well as trying to incorporate a mechanism for long-term memory over phrases. Using Convolutional layers for obtaining better structural representations of the input and at the attention stage could also be beneficial, and can be a direction for further investigation.

6. References

[1] Rajapurkar, P., Zhang, J., Lopyrev, K., Liang, P. *SQuAD: 100,000+ Questions for Machine Comprehension of Text* (2016)

[2] Pennington, J., Socher, R., Manning, C. D. *GloVe: Global Vectors for Word Representation* (2014)

[3] Weston, J., Bordes, A., Chopra, S., Rush, A. M., van Merrinboer, B., Joulin A., Mikolov, T. *Towards AI Complete Question Answering: A Set of Prerequisite Toy Tasks* (2015)

[4] Hermann, K. M., Koisk, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., Blunsom, P. *Teaching Machines to Read and Comprehend* (2015)

[5] Bahdanau, D., Cho, K., Bengio, Y. *Neural Machine Translation By Jointly Learning to Align and Translate* (2015)

[6] Seo, M., Kembhavi, A., Farhadi, A., Hajishirzi, H. *Bi-Directional Attention Flow for Machine Comprehension* (2017)

[7] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Lukasz, K., Polosukhin, I. *Attention is All You Need* (2017)

[8] Xiong, C., Zhong, V., Socher, R. *Dynamic Coattention Network for Question Answering* (2017)

[9] Kumar, A., Ondruska, P., Iyyer, M., Bradbury, J., Gulrajani, I., Zhong, V., Paulus, R., Socher, P. *Ask me Anything: Dynamic Memory Networks for Natural Language Processing* (2016)

[10] Wang, S., Jiang, J. *Machine Comprehension using Match-LSTM and Answer Pointer* (2017)

[11] Vinyals, O., Fortunato, M., Jaitly, N. *Pointer Networks* (2015)

[12] Yang, Z., Dhingra, B., Yuan, Y., Hu, J., Cohen, W. W., Salakhutdinov, R. *Words or Characters? Fine-grained Gating for Reading Comprehension* (2017)

[13] Wang, W., Yang, N., Wei, F., Chang, B., Zhou, M. *Gated Self-Matching Networks for Reading Comprehension and Question Answering* (2017)

[14] Fan, T., Yang, M., Yu, C. *Co-Attention with Answer-Pointer for SQuAD Reading Comprehension Task* (2017)

[15] Hochreiter, S., Schmidhuber, J. *Long Short-Term Memory* (1997)

[16] Cho, K., van Merrienboer, B., Calgar, G., Bahdanau, D., Fethi, B., Holger, S., Bengio, Y. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation* (2014)