A simple sequence attention model for machine comprehension

Marcello Hasegawa

Abstract

Machine comprehension is an important NLP problem with a number of important applications. Particularly question answering has been attracting a lot of attention on the applied research area. This work explores a simple sequence attention architecture for question answering. In this work the Stanford Question and Answering Dataset (SQuAD) introduced by Rajpurkar et al. (2016) is employed.

1 Introduction

Machine comprehension is an important NLP problem with a number of important applications. Particularly question answering has been attracting a lot of attention on the applied research area. This work explores a simple sequence attention architecture to address the question answering problem. In this work the Stanford Question and Answering Dataset (SQuAD) introduced by Rajpurkar et al. (2016) is employed. The SQuAD is machine reading comprehension dataset consisting of 100,000+ questions posed by crowdworkers on a set of Wikipedia articles.

Part of the goals for this work was to be able to provide a solution under limited time and computational resources constraints, such that it could be suitable for a quick introduction to the area of machine comprehension as an application to the question answering problem. In this work a number of architectures were explored but one in particular could balance simplicity and relative performance.

Question answering typically is posed in three different ways based on how the answer can be formulated: open answer, were the task is to generate an answer in natural language; multiple choices, were the task is to predict the right choice and answer extraction, were the task is to extract the answer from a paragraph or body of text. This work focus on the last type of task which aligns to the purpose of the SQuAD dataset. Next the model is described, followed by considerations on the training as well as the results obtained. Important points on other models explored and future steps are highlighted.

2 Data and Pre-processing

The data consists of triples context-question-answer index span (Initial and final indices of the answer within the context passage). The data was tokenized with a basic tokenizer and the Stanford GloVe [1] was used as trained word embeddings. To ensure a larger coverage among the extracted tokens the larger ”Common Crawl” with 840B tokens, 2.2M vocab and 300d vectors embeddings were used. The data was divided in three sets: 87K triples as training set and 8K and 2K triples as validation and test sets respectively. The context and the question were converted to token ids indexed after the entire vocabulary and padded with zeros at a minibatch level. The figure 1 shows a typical context passage containing a highlighted answer for the question: How long did Project Apollo run? Note that there’s more than one possible answer. In this case ”1961 to 1972” or ”Apollo ran from 1961 to 1972”.

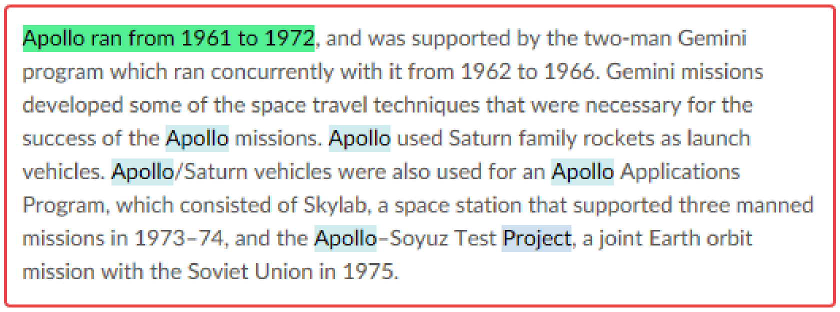
Apollo ran from 1961 to 1972, and was supported by the two-man Gemini program which ran concurrently with it from 1962 to 1966. Gemini missions developed some of the space travel techniques that were necessary for the success of the Apollo missions. Apollo used Saturn family rockets as launch vehicles. Apollo/Saturn vehicles were also used for an Apollo Applications Program, which consisted of Skylab, a space station that supported three manned missions in 1973–74, and the Apollo–Soyuz Test Project, a joint Earth orbit mission with the Soviet Union in 1975.

Figure 1: Typical paragraph or context containing a highlighted answer for the question: How long did Project Apollo run?

3 Model

The model consists of a representation layer, an attention mechanism and a prediction layer as shown in figure 2. The purpose of the representation layer is to generate a representation of the embedding vectors taking into consideration contextual information from each time step for the context and the question as d-dimensional vectors, where d is the size of the RNNs cell states.

For this model two separated GRU networks were employed as representation layer:

$$C = GRU(C_e), Q = GRU(Q_e) \tag{1}$$

were C_e and Q_e are the matrices of the embeddings for each token on the context and the question, respectively. LSTMs were also evaluated producing similar results but the GRUs were preferred due to its lower cost and similar performance. Also in more sophisticated models, bi-directional networks are typically employed. In this work to reduce cost, a single sequence model was adopted resulting in faster training time versus a small performance loss in predictive power.

The obtained representations for the context C and question Q are then used to obtain a normalized affinity matrix $A = softmax(CQ^T)$ which contains the affinity scores between all pairs of context tokens and question tokens. The affinity matrix is then used together with the question representation Q to produce the attention contexts $C^C = AQ$ for the context passage given the question tokens. The attention contexts C^C are combined with the context representation C for each token and sent through a linear projection layer.

The resulting output is then sent to another linear projection operation (not represented in figure 2) and is normalized into a vector representing the probability $p(a_s|C,Q)$ of each token being the initial position or ”start” of the answer within the context passage. The projection layer result is also feed into another GRU network which has the role of propagating the temporal information into the context vector projected onto the attention context. The resulting vectors are sent to a linear projection and then into a softmax to give the probability $p(a_e|C,Q)$ of each token being the final position or ”end” of the answer within the context passage. To train the model a cross-entropy loss is minimized based on the training examples.

4 Training and Results

To validate the model implementation and experiment with the model behavior a small sample of the actual training set was used. The model was observed to easily overfit the small training set. During the model tuning, the following parameters were considered: size of GRU cells internal state, gradient clipping threshold, dropout keep rate and the optimizer (SGD, Adam, Proximal Adagrad). The model was able to generalize with F1 score at 27.1 and EM score at 16.8 [1] , refer to Rajpurkar et al. (2016) for definitions. These numbers are still low compared to results from more sophisticated

[1]Metrics calculated on the 2K test set at a token id level.

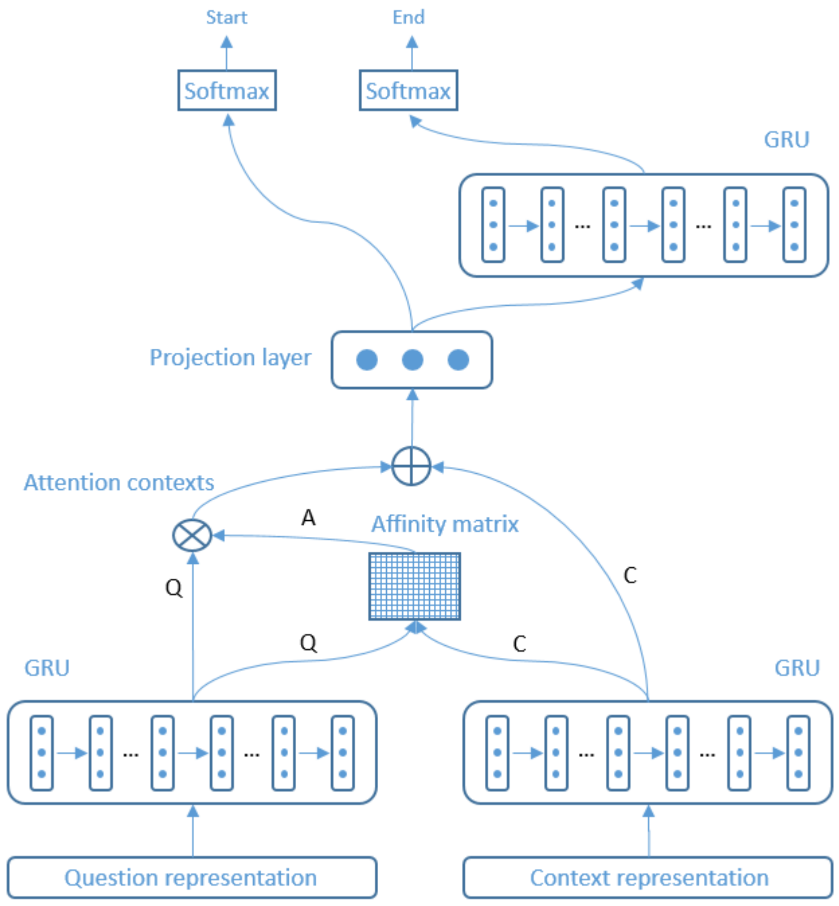

Figure 2: Simple sequence attention model for question answering. The model consists of a representation layer an attention mechanism and a prediction layer.

models or models which went through a more extensive training and tuning process. Also at this phase, small changes in the model architecture were experimented with. In particular, a comparison between fixing the embeddings layer versus letting the model learn the embeddings was made. It was verified that with the model learning the embeddings, it could easily overfit the data, while with fixed embeddings the learning rate was very small. The best result was obtained with embeddings learning at a higher dropout rate and early stopping.

5 Other Models and Future Steps

A number of simple and also more complex architectures were also implemented and evaluated. A simple model based on Chen et al. (2016) employed a GRU network fed with the question embeddings and a single vector representing the question was taken from the internal state of the last time step. Another GRU network produced a representation for the context and a simple dot product between the representation for each context token and the question vector was sent through two softmax units representing the ”start” and ”end” probabilities. These probabilities were then maximized during the training phase. This model performed well on the small training set evaluation but showed less generalization power on the full set. Another model implemented was the model described in Wang et al. (2016) this model consists of a preprocessing layer, a match-LSTM layer and an answer pointer layer. The details of the model are extensively described in the reference. The resulting model had a much longer training time than the models previously described and the validation and tuning had proven a challenge within the time constraint goals. An interesting idea also explored was to combine the simple encoder of the model based on Chen et al. (2016) and instead of the dot product normalized into probabilities, the question vector was used to weight the context vector before sending it to a pointer network as in Wang et al. (2016). This model was implemented and the initial checkings completed. This could be an alternative model of less complexity while carrying components of more sophisticated models and can be further pursued as a future work.

6 Conclusion

In this work a simple sequence attention model for question answering on the SQuAD dataset was presented. Besides this model a few other architectures were explored. The results indicated that this simple model can be used to demonstrate the learning power of attention models for question answering within the scope of a small project.

References

[1] https://nlp.stanford.edu/projects/glove/

[2] Rajpurkar, P. & Zhang, J. & Lopyrev, K. & Liang, P. (2016) Squad: 100,000+ questions for machine comprehension of text. *arXiv: 1608.07905*.

[3] Xiong, C. & Zhong, V.& Socher, R. (2017) Dynamic coattention networks for question answering. *arXiv: 1611.01604*.

[4] Chen, D. & Bolton, J. & Manning C.D. (2016) A Thorough Examination of the CNN/Daily Main Reading Comprehension Task. *arXiv: 1611.01604*.

[5] Wang, S. & Jiang, J. (2016) Machine Comprehension using Match-LSTM and Answer Pointer. *arXiv: 1608.07905*.

[6] Wang, Z. & Mi, H. & Hamza, W. & Florian R. (2016) Multi-Perspective Context Matching for Machine Comprehension. *arXiv: 1612.04211*.