Deep QA models for SQUAD

Divyendra M
Computer Science Department
University of Massachusetts Amherst
dmikkilineni@umass.edu

Gadde Venkata Sai Kumar
Computer Science Department
University of Massachusetts Amherst
gvenkatasaik@umass.edu

Abstract

This document contains the final report for the Neural Networks project. We study the challenge of closed-domain question answering and build neural models with a focus on answering questions in SQUAD dataset. We start with basic forms of confining question and the context onto a single hidden state representation and later, realizing their poor performances due to information bottlenecks, implement a model with timestep outputs and attention mechanisms, BIDAF to achieve massive accuracy gains. We analyse the statistics of the question for which we were predicting successfully and not.

1. Introduction

In this project, we explore the challenges of building a QA model by implementing vanilla sequence neural models and compare them with models with more information flow and attention mechanism.

A typical reading comprehension task involves reading the question at hand, analyze the corresponding comprehension paragraph for context and predict an answer sentence within the context paragraph. And hence this task behoves the neural models to represent the context and the question in it's state space efficiently and make temporal, contextual dependencies.

Traditional approaches tackling QA challenges(Especially Factoid QA) are mainly Information retrieval driven and models based on syntactic matching approaches. Appreciating the neural models recently shown ability to encode the semantic properties of the language, a lot of neural models, techniques have been proposed to tackle the QA domain. Inspired by this work, we first approach the problem using a vanilla state encoding neural models to compress the question context in a single representation. With this model performing poorly, we move ahead to implement RNN sequence models with sequential representations rather than single state representations, and bidirectional attention between these sequential representations of question and context as proposed in BIDAF to achieve high predictive performances.

2. Dataset

For a supervised approach of building the neural QA model, we woule be in need of large amounts of xamples in the form of (Question, Context, AnswerSpan), where the Answer span could be the start end index pointing the answer in the given Context para. We use the recently published SQUAD [5] dataset for this purpose

SQUAD dataset consists of 81403 question-answer pairs, each with a context paragraph. Unlike synthetically generated datasets like bAbI[[8]],the questions in this dataset are generated by humans based on Wikipedia articles. This is more realistic and poses a higher difficulty in inference capabilities. Below is an example entry from the dataset:
Question: Why was Tesla returned to Gospic?
Context Paragraph: On 24 March 1879, Tesla was returned to Gospic under police guard for not having a residence permit. On 17 April 1879, Milutin Tesla died at the age of 60 after contracting an unspecified illness (although some sources say that he died of a stroke). During that year, Tesla taught a large class of students in his old school, Higher Real Gymanasium, in Gospic.
Answer: [15, 19]: not having a residence permit.

3. Evaluation

We use two widely used metrics to evaluate the QA model:

1. *Exact Match:* Measures the percentage of predictions that match one of the ground truth answers exactly.
2. *F1 Score:* Loosely measures the average overlap between the prediction and the ground truth answer.

4. Related Work

Initially CNNs[7] were used to model question and context taking advantage of CNN properties to model local

connectivity. This approach came down to scan for sentence similarity using relational information given by match words between question and answer pairs. But tasks like Reading Comprehension need more than local connectivity and model sequential information capturing long term dependencies which CNNs are not structurally well suited for. Much of the later work favoured using RNNs to model sentences which have the properties of sentence summarization to fixed vector representation. Another suitability of their usage is their ability to model variable length sequences.
But practically RNNS had trouble capturing the entire information in the sentence. So RNNs were augmented using Attention technique, first introduced by [2], where we try to capture the important parts of the context paragraph wrt to the given question. This techniques has seen much use in a wide array of applications[[2]][[3]].
In this project, we eventually implement a version of Bidirectional Attention Flow[[6]] where the is a two way attention capturing mechanism:from Context to Query; from Query to Context. The paper's implementation has achieved state of the art results in SQUAD dataset

5. Approach

In this section, we detail the QA system and it's internal module architectures we implemented. First, we introduce the required terminology used. Second, we brief about our initial 'single state summarization' approaches using RNN architectures. Finally, we describe the multi-output representations and attention mechanism we eventually adopted as stated in BIDAF paper[[6]]

5.1. Terminology

From a bird's eye point of view, our system consists of three modules: Question Module; Context Module; Answer Module- the question and context module are independent modules responsible for representing the given question 'q' and context 'x' into embedding representations 'u' and 'h' respectively which are given as an input to the answer module. The answer module with this information tries to predict scores over which word in the context paragraph could be a start index and end index. So the answer module would have two prediction scores as an output:one for the start index, other for end index. This can be seen in Figure:[1]

The given question and answer are represented as a sequence of words: Question 'q' of length 'J', $q = q_1, q_2, .., q_J$ and Context 'x' of length 'T', $x = x_1, x_2, .., x_T$. Each of the word in 'q' and 'x' is in turn a semantic vector representation in itself for which we have used the GLOVE[[4]] 100-sized vectors. The overall model tries to map these inputs q, x to output a span- $f : (q, x) -> (start, end)$ such that $x_1 < start < end < x_T$. However, this function mapping is implemented by the following two different approaches.

5.2. Summarization Approach

RNNs are known to represent/summarize a entire sentence into a single hidden embedding representation. With the same motivation, we have implemented our question and answer modules to summarize given question 'q' and input 'x' into single state vector representations of equal sizes-'d'.
We have tried three different RNN encoding procedures:

- Vanilla RNN
- LSTM
- Bidirectional LSTM.

While the first two cell representations encode only left to right temporal relations, the third variants outputs two representations, one for Left-Right relations, other for Right-Left relations, both of which are concatenated to give state representation of size '2d'. For question module it would be:
$QuestionModule : (q_1, q_2, .., q_J) -> (u1, u2) -> (u : 2d)$
The Context Module encodes its the context conditioned on the question's embedding representation. Intuitively it encodes both of the question and state into a single vector output representation
$ContextModule : (x_1, x_2, .., x_T; u) -> (h)$
The Answer module is a fully connected regressor which projects its inputs representations u, h into two different scores which are softmaxed to give two vectors of prediction over start index and end index. Obviously, each of the prediction vectors are of size 'T', the size of the context:
$AnswerModule : (u : 1 * d, h : 1 * d) -> (s : 1 * T, e : 1 * T)$
Figure-1 gives the pictorial representation of our entire initial setup of using trying to predict the span using single state summarization approach

5.3. Bi-Directional Attention Model

Our initial setup of summarization performed poorly as mentioned in Table [1]. We surmise it is because of the informational bottleneck present in these summarization approaches and the additional gradient flow problems in such large sequence single-path networks. Many neural architectures like Residual nets, Dense nets have been proposed to achieve multiple ambits of information flow and gradient passages. With a similar motivation, RNNs are modeled to have sequential representations allowing each word to represent it's information as an output embedding. This is contradiction to our initial approach of summarization. So the RNN processing the input passes along information

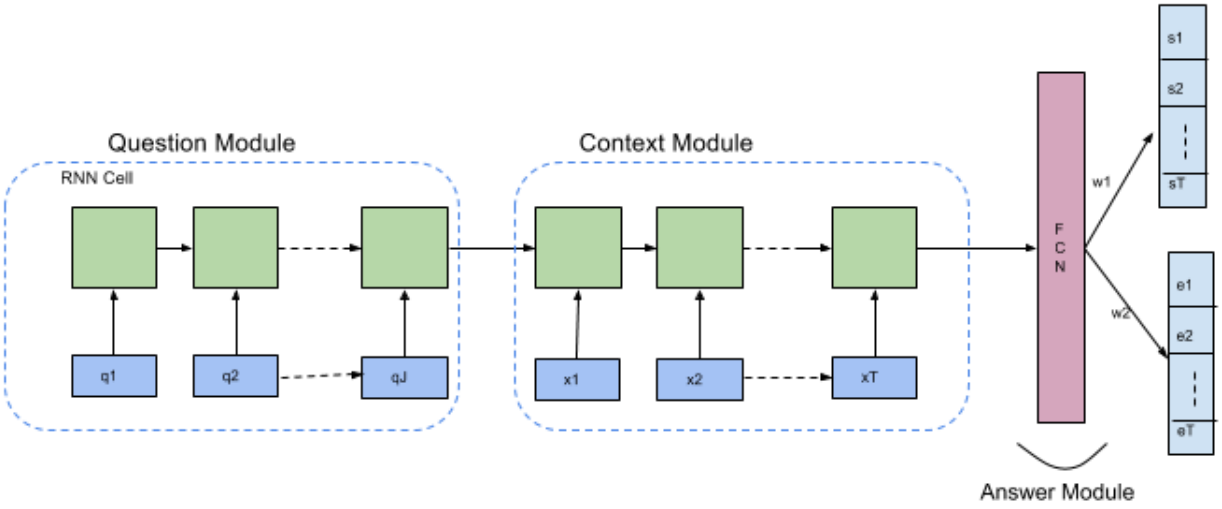

Figure 1: Initial setup using Summarization into a single vector representation and using a FullyConnected Regressor on it

about each word it sees which is given as an input to an Attention module which passes assigns weights to each word embedding and gives this input to the Answer module. The Answer module is another RNN model which takes the attention weighted answer embeddings at each timestep and predicts the start and end span indices. This Answer module is different from our initial setup where it was Fully connected regressor.

5.3.1 Question and Answer Module

The original BIDAF model used a hierarchical input representation where there are three levels of input representations:Character level, Word level, Contextual level. We just confine our model to take the word embeddings of each word as an input to allow the Bi-LSTM to represent that input word as corresponding output vector of size (1*2d).
$QuestionModule : (q_1, q_2, .., q_T) -> (u_1, u_2, .., u_J)$
$ContextModule : (x_1, x_2, .., x_T) -> (h_1, h_2, .., h_T)$
Each of these outputs can be considered as contextual embeddings of the words where the word is represented in a 'd' dimensional with respect to the surrounding context words.

5.3.2 Attention Module

This is an additional module we introduce as done in BIDAF model. This module couples the query and context input embedding vectors and produces a set of query-aware feature embeddings for each word in the context. It constructs two kinds of attention(bi-directional):

- **Context to Query attention**
  Signifies important query information for every word in the context
- **Query to Context attention**
  Signifies the context information have the closest similarity to the *entire* query vector

We first construct a similarity matrix 'S'(T, J) which embodies the similarity between every question word and context word. It is a simple dotproduct distance measure. From 'S' we will do a softmax across 'J'(all query words) to get the weights with which we do a weighted average of the question vectors for each word step x_t of context. This weighted attention query for each context word is represented as $\widetilde{U}(2d, T)$. This would be the Context to Query Attention matrix. $\widetilde{U} = (Softmax(S).H)$
On the other hand, for constructing Query to Context Attention matrix,$\widetilde{H}(2d, T)$, we first construct a $\widetilde{h}(1, 2d) = \Sigma_t b_t.H_{:t}$ where b constitutes attention weights for each context word: $b = Softmax(Max_{col}(S))$. To use this same attention weighted context vector across all timesteps, we tile $\widetilde{h}$ 'T' times to get our $\widetilde{H}$.
Now as suggested in the BIDAF model we will concat vectors of both these attention matrices along with context outputs from the Context module to give the Attention module's final output A:(T, 8d). $A_t = [h_t \bigoplus \widetilde{u_t} \bigoplus (h \bigodot \widetilde{u}) \bigoplus (h \bigodot \widetilde{h})]$ where $\bigoplus \bigodot$ refer to vector concatenation and Hadamard product respectively. This 'A' is sent as an input to out Answer Module.

5.3.3 Answer Module

The answer module is a bi-directional LSTM which tries to model that bi-directional attention weighted context embed-

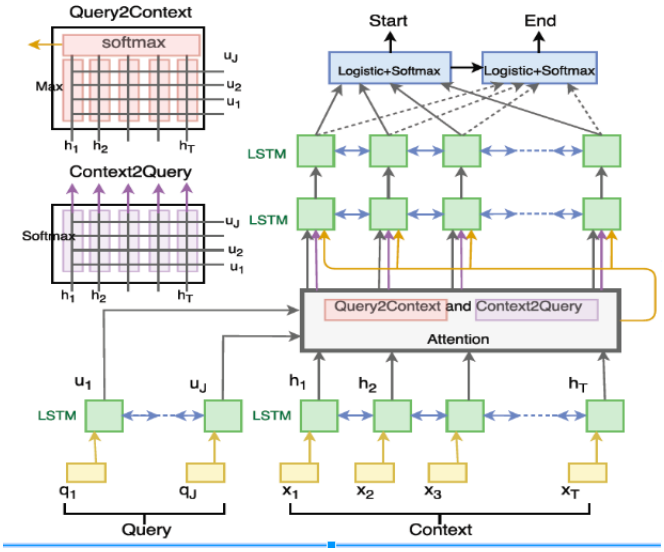

Figure 2: BIDAF architecture with only contextual embedding input representation.(Adopted from[[6]]

dings and emit outputs for each of this word. These ouputs are then projected and softmaxed to give probabilities for the start index and end index.

6. Experiments

As explained above initially we implemented a RNN based summarization method.

We used a Vanilla RNN cell as a start. Considering their fair known problems with exploding and vanishing gradients, we have used gradient clipping technique with a max grad norm of '10'.
But even then this cells are known to have long term dependency problems, then we experimented with GRU cells which is capable of learning long term dependencies and gradient flows. We preferred GRUs over LSTMS considering that former involve less computation and are easy to train..
Then, we updated our summarization model to use Bi-LSTM cells to model dependencies in both directions. This increased our hidden state representation from 'd' to '2d'.
We have used multiple hidden state representations [24, 64, 512] and cell types but none of the above summarization techniques performed even marginally well as shown in the results(Table.1). We opine the reasons of its poor performance in the Approach section.

The below techniques are some of them which we employed for both our Summarization approach and BIDAF approach:

Figure 3: F1 score

1. Since both start index probabilty distribution and end index distributions are considered independent here, there were many cases where we chose an maximum end index after the start index position. Hard programming in the answer module to eliminate such choices improved out EM scores by a factor of '1'.

2. All the out of vocabulary words not found in the GLOVE library are initialized with random vectors as done usually in practice

3. The questions have an average length of 20 words and the context paragraph have average length 500. So we zero-padded all the question and context sentences to 20 and 500 respectively. But to avoid additional RNN cell computations for these zero padded words, we have used DynamicRNNs feature in Tensorflow which simply does an Identity operation for these padded lengths. This is more for correctness sake than for computational efficiency.

We then implemented out BIDAF model as explained in the Approach section by introducing an extra additional module. Below are the model parameters we settled for:

1. learninng rate: 0.0005

2. Dropout: 0.3

3. Epochs: 7

The results and significant accuracy boost using this model can be seen in Table [1].

Avoid Over-fitting As can be observed above, the model has generalized well when the dropout is used.

Table 1: Results for the different models

S.No	Method	F1Score	EM
1.	Baseline	11.704	4
2.	Baseline(GRU)	17.102	6
3.	Baseline(Bidir LSTM)	19.207	6
4.	Attention Model	72.72	60
5.	Attention Model (Dropout=0.3)	69.66	56
6.	R-Net (State of the art)	88.126	82

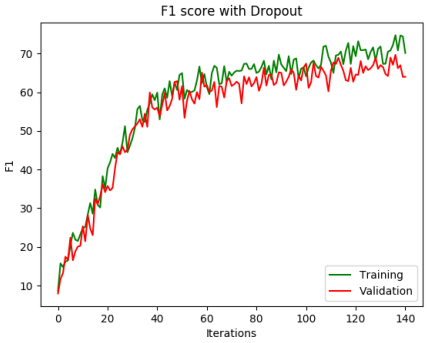

Figure 4: F1 score with dropout = 0.3

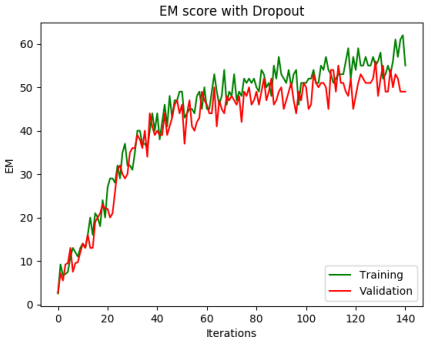

Figure 6: Exact match with dropout = 0.3

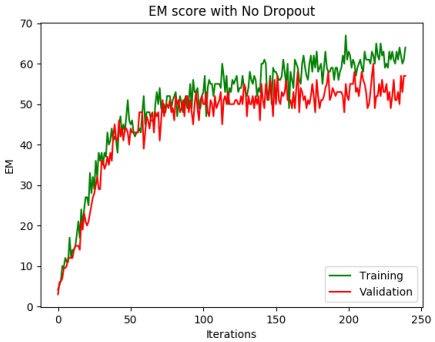

Figure 5: Exact Match with dropout = 0

7. Conclusion

The SQUAD dataset is a very complex dataset, as it involved lot of feature engineering and hand tweaking of different parameters. We found that the model is not robust to small changes in the architectures like removing the final LSTM layer or some operations in the attention layer. We would like to extend this project to include dependency parsing for the model in order to decrease the accuracy.

7.1. References

List and number all bibliographical references in 9-point Times, single-spaced, at the end of your paper. When referenced in the text, enclose the citation number in square brackets, for example [1]. Where appropriate, include the name(s) of editors of referenced books.

References

[1] Authors. The frobnicatable foo filter, 2014. Face and Gesture submission ID 324. Supplied as additional material `fg324.pdf`.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals. Listen, attend and spell. *arXiv preprint arXiv:1508.01211*, 2015.

[4] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[5] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[6] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

[7] A. Severyn and A. Moschitti. Modeling relational information in question-answer pairs with convolutional neural networks. *arXiv preprint arXiv:1604.01178*, 2016.

[8] J. Weston, A. Bordes, S. Chopra, A. M. Rush, B. van Merriënboer, A. Joulin, and T. Mikolov. Towards ai-complete

question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.