Fill in the ___: A Deep Learning Investigation of Machine Comprehension

Brandon Lin, Yonah Mann, Rohan Menezes

Abstract

Cloze-style reading comprehension is a task designed to assess a language model's understanding of textual meaning. In this task, one attempts to fill in the blank of a summary sentence based on a much larger document. We show that deep learning models (specifically, recurrent neural networks) have a significant improvement over base non-deep methods for this language task due to their ability to encode the general sense of a document and capture long range dependencies. We also apply attention methods to basic deep architectures to further strengthen its performance on this task.

1 Introduction

A key part of making a machine understand text is making sure it understands its meaning. Understanding meaning can be difficult because of complex structure of language and a non-concrete way of representing words in language. What is even more difficult is being able to draw inferences from one's understanding; this requires a deeper familiarity with the interplay between entities and actions in a piece of text which is difficult even for humans to do. Therefore, being able to capture this kind of understanding is a very important task in text processing and has ramifications in domains like text summarization, human-computer interaction, etc.

Over time, a lot of focus has been brought to the domain of *reading comprehension*, the task of being able to read a passage of text and pass some assessment of understanding the passage contents. For humans, reading comprehension can be assessed through a reading comprehension exam, where one is presented with a passage and various questions about the passage. Typically, these questions come in the form of multiple choice, where each question is supplemented with a few choices as possible answers.

Many strides have been made in this area, the first of which involved curating a new dataset for this task (Hermann et al., 2015) [1] and running some preliminary models on it. Various attention mechanisms have been proposed to solve this task and improve baseline accuracies. (Kadlec et al., 2016) [2] proposes adding query attention to individual document words through an attention sum network, while (Cui et al., 2016) [3] proposes overlaying multiple layers of attention.

2 Problem Formulation

We formulate this reading comprehension task as a *cloze-style question answering task* in which a learning model is supplied a document as well as a query, which contains a blank placeholder. After also being given a set of choices to fill in the blank, the model then has to determine which choice best fits the blank placeholder.

Formally, a model is given the triple $(\mathbf{D}, \mathbf{Q}, E)$ as training data, containing the document, query, and entity set. The document and query can be written as

$$\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_m]$$
$$\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n]$$

where each $\mathbf{d}_i, \mathbf{q}_j$ are words. The entity set

$$E = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_p\}$$

is a subset of the words in $\mathbf{D}$. $\mathbf{Q}$ also contains

a "blank" word $\mathbf{q}_i$ that has a "correct answer" $\mathbf{e}_j \in \mathbf{E}$.

2.1 Dataset

The dataset we use is the CNN/DailyMail dataset introduced in (Hermann et al., 2015). The data consists of approximiately 380K questions and 90K unique documents. Each document and query has already been tokenized with relevant "confusion" entities extracted from the text. Each entity in the text is replaced with a marker of the form `@entity<n>`, where `n` is a entity-specific unique integer. The query also contains a `@placeholder` marker that the model must fill in.

While the data contains information about what actual entities correspond to the entity markers, we choose to not use this information and make this data anonymous to the model. In this manner, we should not be able to use pre-existing word embeddings to bias our prediction of the correct answer. This is in spirit of the current literature; a model should be able to achieve similar performance to that of the de-anonymized version, so we maintain this notion in our experiment.

2.2 Data Hypotheses

A priori we make a few conjectures about the nature of the data to help guide us in our model building process.

1. Any placeholder entity in a query will be present in the corresponding document. This hypothesis is rather simple and common sense. It simply says that the entity must be present in an article to be included in the corresponding summary of that article.

2. The query alone does not provide any syntactical information on the placeholder. This hypothesis simply states that we cannot learn anything from the sentence structure of the query; the placeholder can be determined by the context of the article alone. This relies on the fact that a placeholder is an entity. In general, proper nouns (e.g. entities) are substitutable for each other without violating grammatical rules. On the other hand, this would not be true for prepositions. It is much more likely to say "in the box" as opposed to "through the box." However, "in the Super Bowl" and "in the Champions League Final" are both equally likely and gramatically correct statements.

3. There are mostly long range dependencies present between the main entities of the document. This is our most controversial data hypothesis. Some might claim that most articles contain summary sentences that perfectly sum up the main relationships between the entities of the article (and thus, that neighboring words in the query will be neighboring words in the document). However, while this may be true in some cases, the main entities of the article would occur in many, many places besides the summary sentence. Thus, there would be no way for the model to differentiate which was the summary sentence without a broader notion of context over the whole document.

   Furthermore, articles generally break down into separate sections that each cover a different main entity of the article. For example, an article on a sports championship game might first discuss the action in the game itself, then discuss the two teams, before finally considering the broader implications for the sport as a whole. However, putting all these disparate sections together to understand how they interact and fit together requires understanding these dependencies across a long range.

4. The query sentences are central to the larger text. The queries are all taken from the bullets that CNN and Daily Mail place in their articles to summarize the article for the reader. Thus, we assume that each query, and more importantly the entities in each query, is of central importance to the article.

3 Methods

We employ three different levels of models: a non-deep benchmark, a base deep learning model, and an advanced deep model.

3.1 Non-deep Benchmark

Following is a logistic regression method to solve this problem. We cannot directly apply logistic regression to this problem with the entities as the classes since between examples, the number of entities varies and the "correct class" will vary across examples as well.

We convert each document-query-entity triple $(\mathbf{D}, \mathbf{Q}, E)$ into a set of $|E|$ auxiliary training examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{|E|}$ as follows: $\mathbf{x}_i = f(\mathbf{D}, \mathbf{Q}, e_i)$ will be a "featurization" function of the document, query, and the corresponding entity. y_i is set to 1 if e_i is the "correct" entity for this triple, and 0 otherwise.

Once all these auxiliary training examples are created, we learn a weight vector $\mathbf{w}$ using logistic regression. Recall that logistic regression associates a probability $\eta(\mathbf{x})$ to each training example based on the following formula:

$$\eta(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^\top \mathbf{x}}}$$

During test time, the document-query-entity triple will be converted into the auxiliary training examples as before, and the correct entity is predicted as such:

$$\hat{e} = \arg\max_{e_i \in E} \eta(\mathbf{x}_i)$$

The featurization function captures a variety of features:

- The number of times the entity e appears in the document $\mathbf{D}$.
- The position of the first appearance of e in $\mathbf{D}$.
- Whether there is an n-gram match between the document and query, in the vicinity of the `@placeholder` marker, if the marker is replaced with the entity e.

In essence, this is rather similar to the linear classifier in (Chen et al., 2016) [4]; there, they choose their weight vector $\mathbf{w}$ to maximize the dot product $\mathbf{w}^\top \mathbf{x}_i$. However, they employ an implementation of *LambdaMART* to find this weight vector. We choose to use logistic regression in ours since it accomplishes the same goal, and can be efficiently run in PyTorch.

3.2 Base Deep Models

The base deep models we consider are BiLSTMs. LSTMs have been proven to be a natural extension of RNNs in the literature, incorporating a forget gate and cell state to choose to remember certain words throughout a sentence. Bidirectional LSTMs allow us to capture forwards as well as backwards long-distance relationships within a sentence. Figure 1 shows how we input each example into the LSTM.

Let N be the total number of entities. Our model begins by concatenating the document $\mathbf{D}$ and the query $\mathbf{Q}$ into a new vector

$$\mathbf{D} \circ \mathbf{Q} = [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_m, |||, \mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n]$$

where $|||$ represents a delimiter token. We feed this vector into the LSTM to yield a final output vector $\mathbf{y} \in \mathbb{R}^x$, where x is the output size. $\mathbf{y}$ is then fed into a fully connected layer to an output vector in $\mathbb{R}^N$. We then select and index the entities corresponding to the document to obtain another vector $v \in \mathbb{R}^{|E|}$, and finally feed this through a softmax and employ a negative log-likelihood loss function.

3.3 Advanced Deep Models: Incorporating Attention

We now consider adding an attention mechanism on top of our base deep models. Specifically, we use a form of double attention, both from the document to the query and from the query to the document. Our model draws inspiration from (Cui et al., 2016) [3], except we expand the flexibility of the network to incorporate bilinear general attention instead of normal attention.

To start, we feed our query $\mathbf{Q}$ and our document $\mathbf{D}$ into separate bidirectional GRUs and ex-

tract the hidden states for each time step, yielding document embeddings $\tilde{\mathbf{d}}_1, \tilde{\mathbf{d}}_2, \ldots, \tilde{\mathbf{d}}_m$ and query embeddings $\tilde{\mathbf{q}}_1, \tilde{\mathbf{q}}_2, \ldots, \tilde{\mathbf{q}}_n$. We then apply attention by constructing a matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$ as follows:

$$M_{ij} = \tilde{\mathbf{d}}_i^\top \mathbf{W} \tilde{\mathbf{q}}_j$$

where $\mathbf{W}$ is a weight matrix that we learn. This, in effect, constructs a matrix with dot products between all pairs of document words and query words. We then take this matrix and separately softmax over both the rows and the columns. This gives us the relative importance of the document words in terms of the query (α) and the query words in terms of the document (β), respectively. Finally, we average over the query words of the beta matrix and output

$$\mathbf{v} = \text{softmax}(\alpha^\top \beta)$$

to get a final probability distribution over all the document words. The model outputs a guess for the blank based solely on which document words were present in the original document.

We specifically chose this form of double attention because it improves upon and pushes further the basic idea behind regular attention. Normally, attention is designed to weight the words of the document in terms of how relevant they are to the query. Then, you would simply be able to take the most relevant entity and output that as the answer to your question. However, this model inherently assumes that each query word is equally important and thus, has an equal say in determining the relevance of a particular document word. This assumption is clearly simplistic and misguided, which is why we introduce double attention. This form of attention first learns the relevance of each query word in relation to the document. Now that the query words have weighting in relation to their importance, we can weight the important of the document words in relation to the query with a more rational sense of the query words' importance in mind.

3.4 Additional Implementation Notes

All word embeddings were retrieved from the Stanford NLP website for pre-trained GloVe embeddings (Pennington et al., 2014) [5]. We use an embedding size of $d = 50$, and anonymized entity and placeholder embeddings were randomly generated according to the distribution $\mathcal{N}(0, 1)$. We choose a hidden size of $h = 128$, and a learning rate of $\eta = 0.001$.

4 Results

We summarize our results in the table below. Figure 2 shows the respective learning curves for training.

Model	Accuracy
Linear Classifier	29.05%
BiLSTM	39.27%
Attention w/ BiLSTM	?%

We see that our deep BiLSTM base model outperforms the logistic regression model by quite a bit–namely, by 10%. This does show that locality is not enough to assume when looking at a word; there are long-range dependencies within the context of a document that have added importance in recognizing an entity within a document. The features of logistic regression involving n-grams and word counts were not sufficient to capture the complexity of the document.

We also found that the LSTM converged at a relatively consistent pace, obtaining a pretty decent loss after approximately 5 epochs. We suspect that we can obtain slightly better results with a few more epochs of training, but also wanted to avoid overfitting the training data.

Our attention model was rather difficult to train, and unfortunately was not able to learn any meaningful results beyond random guessing. We find that our model's learning curve steadily hovered around a constant loss, demonstrating that the model is not learning anything substantial from the data. We suspect that this may be due to either an implementation error or an issue with our architecture. Trying simpler architectures with less attention layers did not seem to be solving the problem, either. We also suspect that this is not an issue with the length of training time, since the learning curve did not seem to be making any meaningful progress within the first few hours of training.

5 Conclusion

Ultimately, the results of our model showed that our initial data hypotheses were indeed correct. While models based just on the query could not do much better than random guessing, logistic regression over the concatenated document and query (data hypothesis #2) that predicts one of the entities present in the document (data hypothesis #1) did much better than random guessing. Because the model architecture was centered around them, clearly, exploiting these two data hypotheses were key to the model's $6\times$ improvement.

However, our deep model went even further by using data hypotheses #3 and #4 by taking into account long range dependencies via an RNN and a central summary of the entire document through the hidden state. Since it also took in both the document and the query and predicted from the entities present in the document, it incorporated all four of our hypotheses and indeed performed the best out of all the models we tried. Thus, we were clearly able to demonstrate the power of deep learning by demonstrating how it can incorporate our more complicated data hypotheses (#3 and 4) to achieve far better results (an almost doubled accuracy).

Unfortunately, our attention RNN also demonstrated another key property of deep learning: that it is extremely hard to train with complex architectures. Although attention allows us to exploit hypotheses #3 and 4 even further, it requires more layers, more mathematical operations, and even some complex shuffling. All of these added bells and whistles made our model very difficult to train and prevented us from accessed these hypothesized benefits.

Future work should be devoted to refining our attention model to achieve a better accuracy than the base deep BiLSTM model. Furthermore, with the long lengths of articles, there is a high likelihood of a vanishing gradient with respect to our complex architecture. Work could be done to prevent this issue by using advanced seq2seq models to transform articles into a smaller size.

6 Acknowledgements

We would like to thank Jeffrey Cheng, David Rolnick, and Konrad Kording for sharing their knowledge through the CIS 700 course offering this past semester. We would also like to thank Erik Zhao for allowing us to borrow his GPU for training and testing our models.

References

[1] K. M. Hermann, T. Koisk, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," 2015.

[2] R. Kadlec, M. Schmid, O. Bajgar, and J. Kleindienst, "Text understanding with the attention sum reader network," 2016.

[3] Y. Cui, Z. Chen, S. Wei, S. Wang, T. Liu, and G. Hu, "Attention-over-attention neural networks for reading comprehension," 2016.

[4] D. Chen, J. Bolton, and C. D. Manning, "A thorough examination of the cnn/daily mail reading comprehension task," 2016.

[5] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

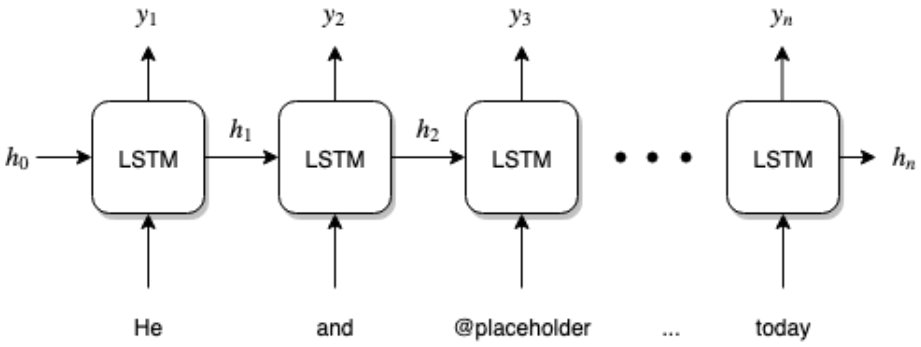

Figure 1: How we pass in a document-query pair into an LSTM.

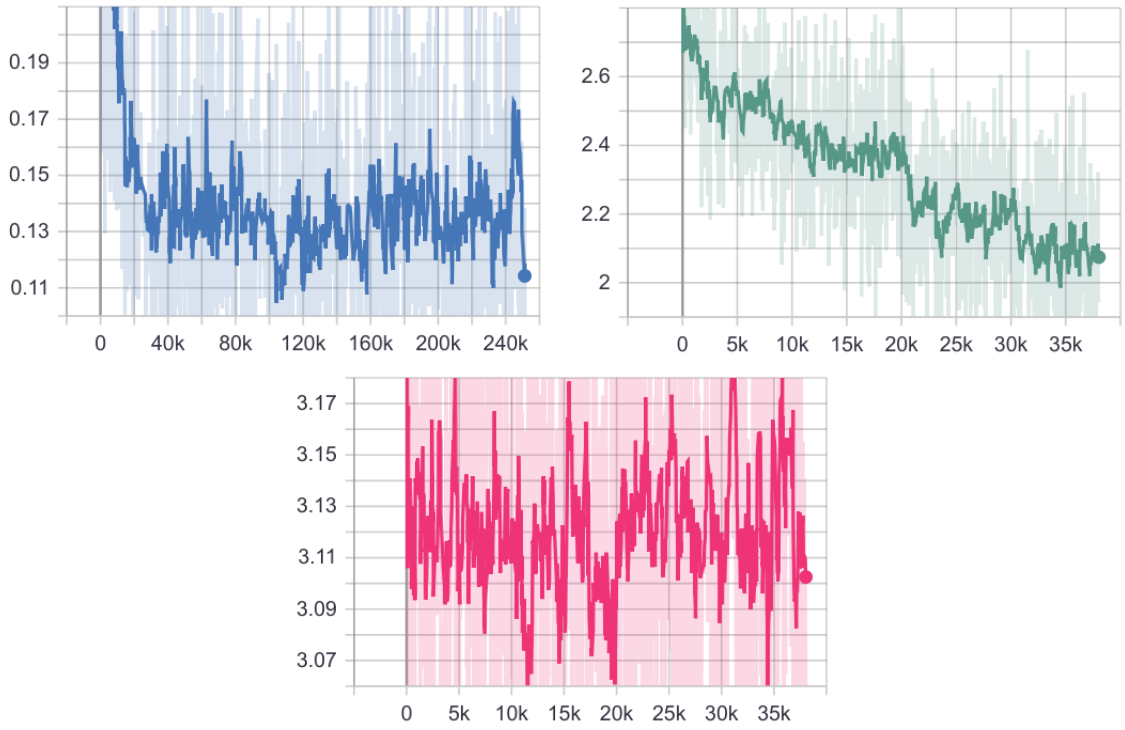

Figure 2: Loss Curves for Logistic Regression, BiLSTM, and BiGRU with Attention