Multi-Hop Question Answering with Bi-Attention Processing and CNNs

Laura Cruz-Albrecht
Department of Computer Science
Stanford University
lcruzalb@stanford.edu

Krishna Patel
Department of Computer Science
Stanford University
kpatel7@stanford.edu

Abstract

Question answering systems are an exciting but challenging application of Natural Language Processing. While much work has been done on general QA, there is a lack of work in the realm of QA requiring multi-hop reasoning, where the QA system has to reason over information from multiple documents to generate an answer. We aimed to create a multi-hop QA model that utilized novel architectures to improve upon the exact match (EM) and F_1 scores of the current publicly available baseline published along with the HotpotQA multi-hop question answering data set. Our best model achieved a QA EM of 40.78 and an F_1 of 53.77, just shy of the published baseline. Through this process, however, we have demonstrated how the use of a 2 dimensional CNN can be approximately comparable to the use of more traditional NLP architectures.

1 Introduction

Question answering as a task is incredibly complex and inherently fascinating, as it asks a model to perform human-like reasoning to identify the answer to a given question. While there is a clear scientific motivation to build an excellent QA system, good question answering models also have the potential to be applied in many domains to increase accessibility of information, efficiency of information retrieval, and can be utilized as a building block for intelligent systems. To date, a substantial amount of research has been conducted in the realm of question answering, specifically concerning single-hop reasoning. Single hop-reasoning, as opposed to multi-hop reasoning, is the ability to identify the answer to a question given a paragraph containing the answer. A popular example of this is the SQuAD data set, in which most questions can be answered by identifying a span within a single sentence [1].

On the other hand, multi-hop reasoning presents a more difficult problem. Multi-hop reasoning requires filtering through multiple paragraphs containing distractions and the aggregation of information across sentences in order to answer the question, which is inherently more difficult than single-hop reasoning as it moves the sphere of question answering research towards computationally modeling what humans have the ability to do when answering questions.

Currently, Yang et al. have published [2] a baseline model (herein referenced as the HotpotQA model) that performs reasonably, while relying on an architecture composed of a chain of recurrent neural networks (RNNs). While the utilization of RNNs is common practice in natural language processing, RNNs are time consuming to train, and lack the ability to focus on local regions by convolving a filter across an input as convolutional neural networks (CNNs) do. These facts generally informed the basis of our approaches, through which we found that 2 dimensional CNNs can be utilized to give approximately a similar performance to the HotpotQA baseline.

2 Related Work

Several prior works have focused on the general task of question-answering, including research on the SQuAD [1], TriviaQA [3], and Google's Natural Questions [4] to name a few. However, the key work most related to our project is "HOTPOTQA: A Dataset for Diverse, Explainable Multi-hop Question Answering" by Yang et al. [2]. This work first introduces a novel type of question-answering dataset, and second, it implements a baseline QA model on this dataset. The dataset contains questions that require multi-hop reasoning, or reasoning with information across multiple documents to obtain the answer. This work is quite innovative, for while much work has previously done to implement question-answering systems, few test the machine's ability to reason across multiple documents.

For their baseline architecture, the authors leverage a model similar to that by Clark and Gardner in "Simple and Effective Multi-Paragraph Reading Comprehension" [5]. Similarly to Yang et al., Clark and Gardner were working with entire documents as input. In particular, they focused on applying the techniques of paragraph-level neural QA models to a document-level systems. They leverage a pipelined method that first extracts which paragraphs from the input document to use for training and testing, then feed these paragraphs through a neural model featuring word and character embeddings, bi-directional GRUs, bi- and self-attention, and a final GRU along with linear layers for answer prediction.

Also related to our work, but in a different vein, is prior work leveraging CNNs for NLP tasks. While convolutional neural networks were originally extensively applied to vision-related tasks, research has found that they can be applied to natural language processing tasks as well. Yin et. al. [6] provide a comparative study of CNNs and RNNs applied to NLP tasks, and find that for some tasks, CNNs can be effectively used in place of RNNs. Further, in "Convolutional Neural Networks for Sentence Classification" by Kim [7], the author illustrates that a simple single-layer CNN with max pooling could be used to classify embedded sentences.

3 Approach

3.1 Existing Baseline

We use the HotpotQA architecture as our baseline. This architecture is designed as illustrated to the right: the context (paragraphs) and question are both transformed into an embedded representation (both word and character level), and fed through a sequence of RNN and linear layers, with bi-attention and self-attention between the first and second RNN layers, respectively.

The model then has an intermediate subroutine to predict whether or not a sentence is a supporting fact for each context sentence. It applies a linear layer to the output of an RNN after self-attention, generating a scalar value corresponding to each sentence, which is then used to then classify that given sentence as a supporting fact. This subroutine obtains an EM score of 22.24 and an F1 score of 66.62 on test.

The output of this classification module is fed back into the main model. The main model then uses 3 RNNs and linear layers to then predict three pieces of information: the start token of the answer, the end token of the answer, and the question type (yes/no/span). The model obtains an overall EM score of 45.46 and F1 score of 58.99 on question answering on the test set.

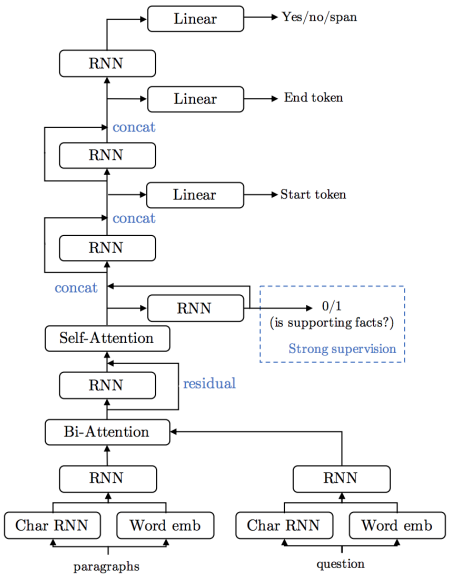

Figure 1: HotpotQA model architecture [2].

3.2 Our Approaches

We experiment with various techniques in an effort to improve upon the HotpotQA baseline.

3.2.1 Supporting Fact Classification Module

As a first approach, we focus specifically on improving supporting fact classification accuracy. As such, we build a module just to classify sentences as supporting facts. In particular, we experiment with using a CNN for this task, rather than the RNN used in the baseline model. While convolutional neural networks have traditionally been utilized more in the realm of computer vision, they have also been applied to natural language processing tasks with great success [7].

Our module utilizes the same first three layers as the HotpotQA baseline (we leverage the publicly available code from [8]), then branches after the bi-attention layer. We first needed to convert the output of the bi-attention layer into a tensor that a convolution could be applied to. The output M of the bi-attention is $(para\_limit, hidden)$, where $para\_limit$ is the (capped) number of words in the context. The context consists of several sentences - we slice M into multiple smaller matrices, such that each contains the words for a single sentence, and stack these slices. This allowed us to convert matrix M into into a new matrix P of dimensions $(sent\_limit, sentence\_length, hidden)$.

We then applied a convolutional layer to tensor P, using the general approach delineated in [7]. Namely, we apply one convolutional layer to embedded input text corresponding to a single sentence, then apply max pooling, and finally apply a linear layer to obtain a scalar output of dimension $(sent\_limit, 1)$. This corresponds to a scalar value for each sentence in the context, and this scalar is the model's classification of the given sentence as a supporting fact.

3.2.2 Processed Bi-Attention Models

Our second approach was to experiment with an end-to-end architecture - namely, to perform question answering (rather than just supporting fact classification). For our model, we again reused the beginning layers of the HotpotQA architecture (embedding and bi-attention). We then branched. In particular, we experimented with manipulating the output of the bi-attention layer to perform the prediction and answering tasks. Then, for the final QA layers, we used the HotpotQA architecture (the final 3 RNNs).

Similarly to what was done in the previously described custom classification module, we sliced the output of the bi-attention layer into a tensor P of dimensions $(sent\_limit, sentence\_length, hidden)$. We then averaged across the words in each sentence, to obtain a tensor of dimensions $(sent\_limit, hidden)$. We then summed across the hidden dimension, to obtain a vector V of dimension $(sent\_limit, 1)$, where each entry v_i in the vector is a scalar that (ideally) would be learned to correspond to whether sentence i is a supporting fact sentence.

This vector V was used as the output vector for deriving the supporting fact classifications. This vector was also then used to rescale the bi-attention output before feeding it through the subsequent layers of the main model.

To rescale, we first applied a sigmoid to V to obtain V^σ, where V_i^σ corresponds to the probability that sentence i is a supporting fact. We then multiplied each row of the bi-attention output (corresponding to a single word w in the context) with V_i^σ, where i is the sentence w belongs to. The goal of rescaling was to amplify the values of words that are part of sentences that the model assigned a higher probability of being a supporting fact. This rescaled output was then fed through an RNN layer; this new output was rescaled in the same way.

We additionally experimented with applying softmax to V instead of sigmoid, but found that sigmoid performed better.

3.2.3 Integrated Supporting Fact CNN Classification Model

After experimenting with an end to end architecture, we realized that supporting fact classification is difficult to optimize when we are optimizing an overall loss of $\mathcal{L} = \mathcal{L}_{qa} + \lambda \mathcal{L}_{sp}$, where $\mathcal{L}_{qa}$ is loss in regards to overall question answering and $\mathcal{L}_{sp}$ is loss in regards to supporting fact classification. Therefore, we proposed creating a pretrained separate module that is optimized for supporting fact

classification only, which would then be integrated into the HotpotQA baseline code in the place of its supporting fact classification architecture.

For our supporting fact classification module, we utilized our first model, described in 3.2.1, which returns scores corresponding to each sentence with dimension $sentence\_limit$, the maximum number of sentences in a given context. In order to reintegrate this vector into the HotpotQA baseline's pathway, we put this vector through an expansion process as follows. We expanded it such that each score for each sentence in the context was repeated x times, where x is the number of words within the sentence whose score it was. We then replaced the -100s that were previously utilized for padding with 0s to create a 0 padded version, and multiplied it across the output of their self-attention layer (whose values correspond to words, not sentences). This reasoning behind this multiplication is that it would result in the augmentation of the signals of words contained in sentences that have been classified as supporting facts, while simultaneously diminishing the signals of sentences that have been classified as not supporting facts.

3.2.4 2D CNN Models

Because we experienced great difficulty in creating a module that effectively classified sentences as supporting facts, our next approaches return back to an end to end model. The first of these two fully integrated models follows a similar architecture to the HotpotQA baseline up until its bi-attention layer (see Figure 1).

The main purpose of the bi-attention layer is to integrate information from the context with information from the question, and therefore, we applied a self-attention layer directly after in order to strengthen the signals of the information before it entered a 2 dimensional CNN layer with max pooling. Although 2 dimensional CNNs are not often utilized in NLP, we were inclined to utilize one because of its capacity to aggregate dependencies across not only the rows but also the columns of our input, which could have the potential to learn useful functionality particularly for multi-hop question answering, since such questions depend on phrases that can be found across relevant sentences. The output of this 2D CNN was integrated into the rest of the HotpotQA question answering architecture, while a copy of its output is run through a 1 dimensional CNN (with identical preprocessing and post processing of inputs and outputs to the model described in 3.2.1) for the purpose of supporting fact classification.

Our last model is similar to the model above, however, the input to the 1 dimensional CNN was instead the output from our bi-attention layer, creating a bifurcation after our bi-attention layer where one pathway heads towards a self-attention layer and the 2D CNN with max pooling, while the other pathway feeds into a 1 dimensional CNN that aggregates the word-based data into sentence-level data (similar to the description in 3.2.1). After passing through the 1D CNN, this information is then reintegrated into the model by passing the sentence-level data outputted by the CNN with dimension $sent\_limit$ into a the same expansion process described in 3.2.3 which results in a supporting fact classification weighted vector in the form of word-level data, which we then multiply element-wise across the output of the 2D CNN. This final output then is passed through the remaining three RNNs of the HotpotQA baseline's question answering architecture.

Q: Mark Levenson toured the country with Stephen Colbert, Paul Dinello, and an actress that played what role in the series "Strangers with Candy"?
Context:
Paragraph 1: Distractor
[1] Wigfield: The Can Do Town That Just May Not is a satirical novel by comedians Amy Sedaris, Paul Dinello, and Stephen Colbert, three of the four creators of the Comedy Central show "Strangers with Candy." [2] It was first published on May 7, 2003 by Hyperion Books.
Paragraph 2: Gold
[3] Amy Louise Sedaris (born March 29, 1961) is an American actress, voice actress, singer, author, screenwriter and comedian. [4] **She is known for playing Jerri Blank in the Comedy Central television series "Strangers with Candy."** [5] She regularly collaborates with her older brother David, a humorist and author. [6] Since 2014, Sedaris has voiced the character Princess Carolyn in the Netflix animated series "BoJack Horseman."
Paragraph 3: Gold
…[6] Levenson composed music for David Sedaris's two Off Broadway shows and numerous recording projects. [7] **He recently toured the country with Stephen Colbert, Amy Sedaris and Paul Dinello in their production of Wigfield, which concluded its run at the U.S. Comedy Arts Festival in Aspen, Colorado.**
Supporting Facts: 4, 7
A: Jerri Blank

Figure 2: A medium level question sampled from the HotpotQA dataset [2].

4 Experiments

4.1 Data

We utilize the HotpotQA [2] dataset for our task of building a model to answer multi-hop questions. Each data point within the dataset consists of a multi-hop question, the answer to the question, 10 paragraphs (or context) to source the answer from (where

2 out of the 10 paragraphs actually contain relevant information), and lists of supporting fact sentence from within the context that support the answer (see Figure 2). The portion of the HotpotQA dataset that we utilized had a total of 89,791 training examples and a total of 7,376 dev examples, which both of which were preprocessed with Yang et al.'s preprocessing script [8].

4.2 Evaluation Methods

Following the pattern of analysis in various QA systems [2], [1], we utilize exact match (EM) and F_1 metrics in order to evaluate the performance of our model. We calculate EM and F_1 metrics on our model's ability to identify supporting facts and our model's ability to identify the correct answers separately. We also utilize Yang et al.'s [2] Joint F1 score and a Joint EM score in order to evaluate the combination of identified supporting facts and answer spans. The Joint F1 score across supporting fact identification and answer spans is defined as follows:

$$P^{(\text{joint})} = P^{(\text{ans})}P^{(\text{sup})}, \quad R^{(\text{joint})} = R^{(\text{ans})}R^{(\text{sup})}, \text{ Joint } F_1 = \frac{2P^{(\text{joint})}R^{(\text{joint})}}{P^{(\text{joint})} + R^{(\text{joint})}}.$$

The Joint EM score across supporting facts and answer spans on supporting facts and answer spans is defined as 1 if both tasks achieve an exact match with the gold answer on a query and 0 otherwise.

4.3 Experiments

For all models, we specified a $para\_limit$ of 2,250, which filters out any data point in the HotpotQA dataset that has greater than 2,250 words in its context. Further, we utilized a supporting fact classification threshold of 0.3, and a supporting fact loss lambda of 1.0, and a maximum sentence length of 100 across all models. The model parameters and hyperparameters used are described below.

Table 1: Model parameters and hyperparameters

Model	epochs	batch size	1D CNN	2D CNN
CNN classif. module	11	24	k: 5, f: 256, s: 1	-
Bi-atten. + sigmoid	6	48	-	-
Integrated SP CNN - SP unit	8	48	k: 5, f: 256, s: 1	-
Integrated SP CNN - full model	2	48	-	-
2D CNN, v1	11	48	k: 5, f: 256, s: 1	k: 5, f: 128, s: 1
2D CNN, v2	5	24	k: 5, f: 256, s: 1	k: 5, f: 128, s: 1

4.4 Results

Table 2: Score comparison

Model	Split	Answer		Sup Fact		Joint		Loss	
		EM	F1	EM	F1	EM	F1	Overall	Sup Fact
HotpotQA baseline	dev	44.44	58.28	21.95	66.66	11.56	40.86	-	-
	test	45.46	58.99	22.24	66.62	12.04	41.37	-	-
CNN classif. module	train	-	-	18.60	60.24	-	-	-	0.092
	dev	-	-	15.85	56.36	-	-	-	0.102
Bi-atten. + sigmoid	train	67.71	74.27	0	9.31	0	6.99	35.38	30.93
	dev	40.19	53.43	0	9.42	0	51.47	39.62	34.54
Integrated SP CNN	train	43.34	50.64	0	0	0	0	6.61	0.19
	dev	32.06	43.20	0	0	0	0	6.38	0.19
2D CNN, v1	train	79.96	85.54	3.49	15.69	2.91	13.87	3.62	0.17
	dev	40.78	53.77	2.59	14.17	1.24	8.30	6.03	0.17
2D CNN, v2	train	18.36	41.37	6.94	28.05	1.61	13.23	7.92	0.15
	dev	8.84	30.44	5.86	26.22	0.90	9.38	9.32	0.15

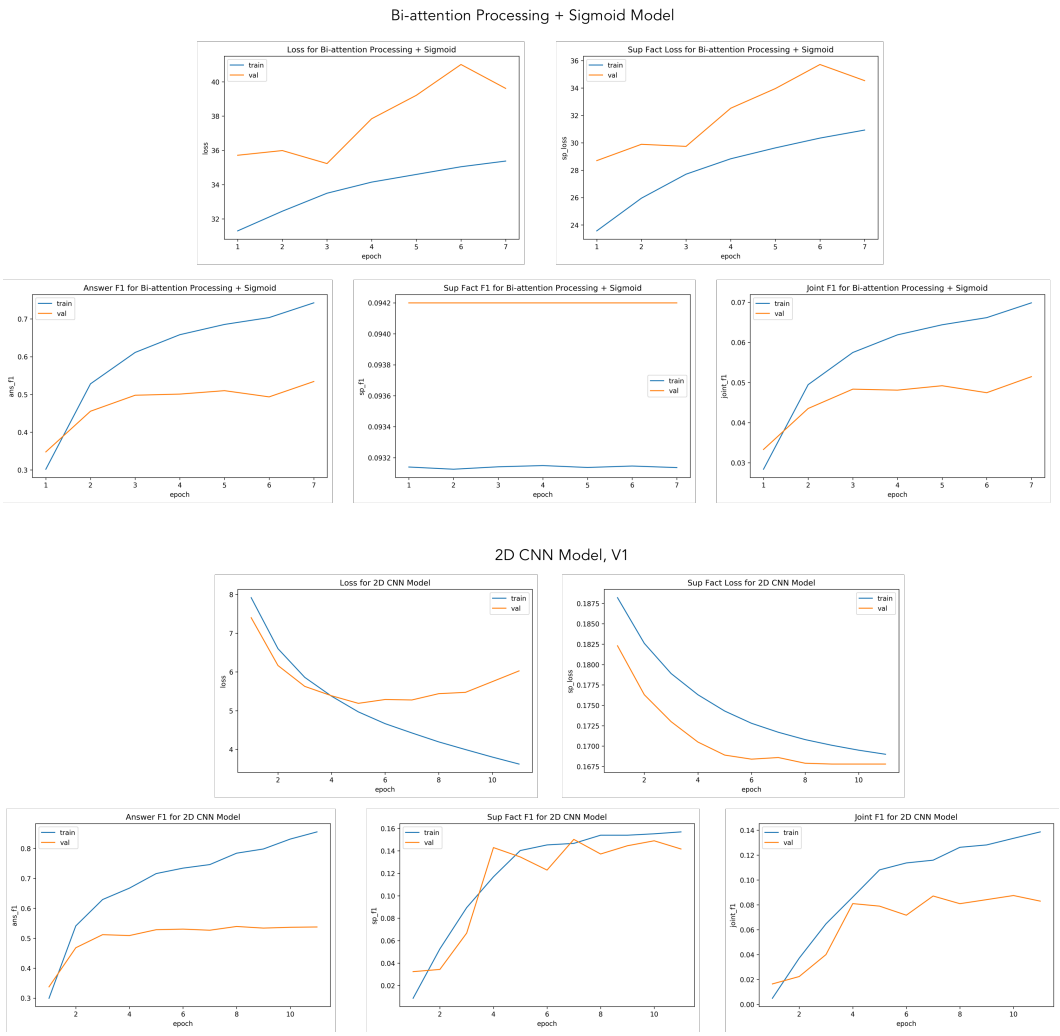

Figure 3: Loss and F1 curves for our two best models.

4.4.1 Supporting Fact Classification Module

We find that the classification module does not beat the baseline. However, it does perform comparably, and we think that the CNN approach is a promising one: perhaps with more hyperparameter tuning (ie, experimenting with different kernel dimensions, strides, and number of filters), as well as with training for longer, it may be possible for the module to attain better performance.

4.4.2 Processed Bi-Attention Models

In these experiments, we find that the overall answer F_1 and EM is comparable (though lower) than the baseline. However, the supporting fact classification performance drops quite a bit. This indicates that the bi-attention rescaling, though promising in principle, is not sufficient to tackle the task at hand. We also find that, applying sigmoid to V performs better than applying softmax. This makes sense, since more than one sentence can be a supporting fact, and sigmoid allows those probabilities to be assigned independently.

4.4.3 Integrated Supporting Fact CNN Classification Model

This model performs very poorly. Rather surprisingly, the supporting fact EM and F_1 drop to 0. The answering part of the model still performs reasonably, with a dev F_1 score of 43.20 (though lower than the baseline, with dev and test around 59), suggesting that the model is simply ignoring the

supporting fact classification and learning to answer questions independent of the attempted strong supervision over supporting facts.

4.4.4 2D CNN Models

We find that the first variant with the 2D CNN performs reasonably on the overall answering task, but does poorly on the supporting fact prediction sub-task. This is perhaps due to the fact that now the model is optimizing an overall loss that only includes the supporting fact classification loss as part of the overall loss; since it's not being optimized directly, in our case we find it's not really optimized at all. The second 2D CNN variant doesn't do well on either account, suggesting that applying the 2D CNN *before* the supporting fact classification (as done in variant 1) is better than after.

5 Analysis

To qualitatively analyze our model, we inspected the answers and supporting facts that our best model (2D CNN, v1) predicted on the dev set. We observed a few key things (see Figure 4). First, we saw that the model got a fair number of answers correct, but without having identified any of the supporting facts. This suggests that while strong supervision over supporting facts would likely be helpful, it is not essential for the model to tackle the task at hand. Second, we observed that most of the extracted supporting facts lists were empty - namely, the model did not classify any of the sentences in the context as supporting facts. While not ideal, this does intuitively make sense given the structure of the model - the supporting fact classification loss is not optimized directly, which may be why the supporting fact classification performance is not prioritized by the model.

Q: Who was known by his stage name Aladin and helped organizations improve their performance as a consultant?
A: Eenasul Fateh
Supporting Facts: [1] Eenasul Fateh (Bengali: ঈনাসুল ফাতেহ; born 3 April 1959), also known by his stage name Aladin, is a Bangladeshi-British cultural practitioner, magician, live artist and former international management consultant. [2] Management consulting is the practice of helping organizations to improve their performance, operating primarily through the analysis of existing organizational problems and the development of plans for improvement.
Model Answer: Eenasul Fateh
Model Supporting Facts: 1, 2

Figure 4.1: Rarely, the model correctly identified both the correct answer and all supporting facts.

Q: The football manager who recruited David Beckham managed Manchester United during what timeframe?
A: from 1986 to 2013
Supporting Facts: [1] Their triumph was made all the more remarkable by the fact that Alex Ferguson had sold experienced players Paul Ince, Mark Hughes and Andrei Kanchelskis before the start of the season, and not made any major signings. [2] Instead, he had drafted in young players like Nicky Butt, David Beckham, Paul Scholes and the Neville brothers, Gary and Phil. [3] Sir Alexander Chapman Ferguson, CBE (born 31 December 1941) is a Scottish former football manager and player who managed Manchester United from 1986 to 2013.
Model Answer: 1986 to 2013
Model Supporting Facts: none identified

Figure 4.2: Often, the model found the correct answer without identifying any supporting facts.

Q: Who was the writer of These Boots Are Made for Walkin' and who died in 2007?
A: Barton Lee Hazlewood
Supporting Facts: [1] "These Boots Are Made for Walkin'" is a hit song written by Lee Hazlewood and recorded by Nancy Sinatra. [2] Barton Lee Hazlewood (July 9, 1929 – August 4, 2007) was an American country and pop singer, songwriter, and record producer, most widely known for his work with guitarist Duane Eddy during the late 1950s and singer Nancy Sinatra in the 1960s.
Model Answer: Nancy Sinatra
Model Supporting Facts: none identified

Figure 4.3: An example of a question that the model failed to find any supporting facts and a correct answer for. [H]

Figure 4: A sampling of our best model's (2D CNN, v1) capability.

6 Conclusion

In conclusion, one key thing that we found was that explicit supporting fact classification is not critical for the ultimate task of question answering in this context. Our models ended up having an answer F_1 score that was much higher than the supporting fact F_1 score and joint F_1 score. This could potentially be attributed to the model still learning how to identify supporting facts implicitly, but lacking the ability to identify them explicitly.

Notably, throughout the course of our experimentation, we also found that the task of optimizing supporting fact classification is not well suited to standard loss calculations. We found this to be the case because in order to classify supporting facts correctly, we are looking to be precise in assigning value to very few sentences out of many. By utilizing a loss function such as cross entropy loss, we will end up minimizing loss by assigning no value to all of the sentences, thereby only incorrectly attaining a few false negatives for the few supporting fact sentences.

Overall, we also found that though we were not able to beat the HotpotQA baseline, we found that CNNs seemed to be a reasonable architectural building block. Our best model (2D CNN, v1), used a 2D CNN across word-level information rather than an RNN, and attained lower but comparable overall Answer F_1 and EM scores.

References

[1] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016.

[2] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[3] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *CoRR*, abs/1705.03551, 2017.

[4] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*, 2019.

[5] Christopher Clark and Matt Gardner. Simple and effective multi-paragraph reading comprehension. *CoRR*, abs/1710.10723, 2017.

[6] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of CNN and RNN for natural language processing. *CoRR*, abs/1702.01923, 2017.

[7] Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

[8] Yang et. al. Hotpotqa github. https://github.com/hotpotqa/hotpot.