AlterNet: Improving Span Conditioning for Convolutional Neural Network Q&A Systems

Stanford CS224N Default Project
Track: SQuAD IID

Simon Camacho
Department of Computer Science
Stanford University
scamacho@stanford.edu

Eva Batelaan
Department of Electrical Engineering
Stanford University
batelaan@stanford.edu

Benjamin Korngiebel
Department of Computer Science
Stanford University
bkorngie@stanford.edu

Abstract

Recent improvements in the field of Q&A have seen a progression towards using convolutional neural networks (CNNs) due to improved training and inference speeds. The QANet model, first introduced in Yu et al. (2018) [1], combines CNNs with self-attention, a concept first seen in the Transformer architecture introduced in Vaswani et al. (2017) [2]. We build upon the BiDAF model described in Seo et al. (2016) [3] to create our own implementation of the QANet model, achieving a single-model F1 score of 65.47, 4.48 points higher than the baseline BiDAF model. We then complement the QANet model with our own extension on the conditional output layer described in Kim and Wolff [4], achieving a dev F1 score of 64.51. Our ensembled model achieves a test F1 score of 63.998.

1 Key Information to include

- Mentor: Kaili Huang
- External Collaborators, Sharing Project: No

2 Introduction

Question-answering (Q&A) is an important sub-field of natural language processing (NLP) that provides a measure for how well NLP systems can understand text, in particular scoring its ability to answer questions by identifying the correct response within a context paragraph. In Q&A systems, the model is typically presented with a piece of text and one or more questions (queries), and then it is expected to derive the answer from the provided context. The answer is some subset of the original context, and some questions may have more than one correct answer. The release of large-annotated datasets, such as The Stanford Question and Answering Dataset 2.0 (SQuAD 2.0) [5], has led to significant advancements in QA over recent years and provided benchmarks on which models can be compared.

Many earlier QA models relied on a sequential end-to-end structure; however, more recent models propose more parallelizable methods. Recent models specifically focus on the use of self-attention, which uses matrix multiplication to determine which words in the context are most relevant to the query. Though the QANet model, Yu et al. combine self-attention with a convolutional neural network (CNN) structure, a model structure used in image processing for its paralleizability, to form a model that efficiently and effectively answers baseline dataset questions [1].

In this paper, we create our own implementation of the QANet model, which achieves an F1 score of 65.47, a significant improvement over the baseline model's score of 60.99. We find that our implementation achieves a similar performance score (61.03 F1) within just an hour of training as opposed to the 2.5 hours required for the baseline. We then extend our implementation of QANet by first implementing the conditional output layer described in Kim and Wolff [4], and then building upon this to create our own conditional output layer. We further experiment with different novel changes on top of our baseline QANet model, including data augmentation, different model ensembling methods, and changing model sizes and hyperparameters.

3 Related Work

Previous state-of-the-art end-to-end machine question-answering systems such as the Bidirectional Attention Flow model designed by Seo et al. (2016) [3] utilize a recurrent model to process sequential inputs using recurrent neural networks (RNNs) in addition to a bidirectional attention mechanism to capture long term dependencies. However, the linear interaction distance between pairs of words makes it difficult for the RNN to learn long-distance dependencies, while restricting words to interacting in the linear order in which they appear. The sequential nature of the model also means that future RNN states cannot be computed in full before past RNN hidden states have been fully computed, drastically slowing down the training and inference time.

Vaswani et al. [2] introduced the transformer with self-attention as a remedy for the issue of slow training times and sequential dependencies. Building on the transformer, Yu et al. (2018) [1] proposed the QANet model, which utilizes convolutional neural networks (CNNs) alongside the self-attention included in the transformer block. The CNNs present in QANet enable the model to learn the local structure of text while self-attention allows the model to determine the global interaction between words, respectively. The parallel nature of the architecture allows the model to significantly decrease training time.

As part of their architecture, most Q&A systems include a task-specific output layer which computes the probabilities of words in a context being the start and end of the answer for a given question. BiDAF and QANet perform a similar technique and independently compute start and end probabilities. Various papers, including Kim and Wolff [4] and Vinyals et al. [6], suggest different methods for conditionally computing end-word probabilities on start-word probabilities. Conditionally computing end probabilities has shown improvement for Q&A models as demonstrated in Kim and Wolff [4]. We explore this, as well as different extensions, in this paper.

4 Approach

4.1 Baselines

For our model baselines, we use three different models in order to track the improvements made by our implementation of and extensions to the QANet architecture: BiDAF, BiDAF + character embeddings, and original QANet. We describe these in the following subsections.

4.1.1 BiDAF

We use a simplified version of the BiDAF model[1] described in Seo et al. (2016) [3], as our most basic code and performance baseline. The BiDAF model has five main layers: the embedding layer, the encoding layer, the context-query attention layer, the model layer, and the output layer. We use this model as a starting point for the rest of our implementations and extensions.

4.1.2 BiDAF + Character Embeddings

For our second baseline, we implemented character embeddings on top of the baseline BiDAF model, as suggested in Yu et al. (2018) [1]. To implement character embeddings, we followed the guidance of Yu et al. (2018) [1]: for an input sequence (i.e. either a context or a query), we feed pre-trained 300-dimensional GloVE [7] character vectors (included in the baseline model) for characters present in the sequence through a two-dimensional convolutional neural network with a kernel size of 5 and

[1]See **here**.

a filter size of 16 (we employ a filter size of 16 because we denote this as $Char_{max}$: the maximum number of characters in a word). We then implement a max pool (with a kernel size of 16) over the output of this convolutional neural network in order to create a final character embedding for each word in the input sequence. We then concatenate these new character embeddings with the pre-trained GLoVe word vectors used in the baseline model and forward these new tensors through the existing two-layer highway network to conclude the embedding layer.

4.1.3 QANet

Using the BiDAF + character embeddings as a base, we implement the QANet structure described in Yu et al. (2018) [1].

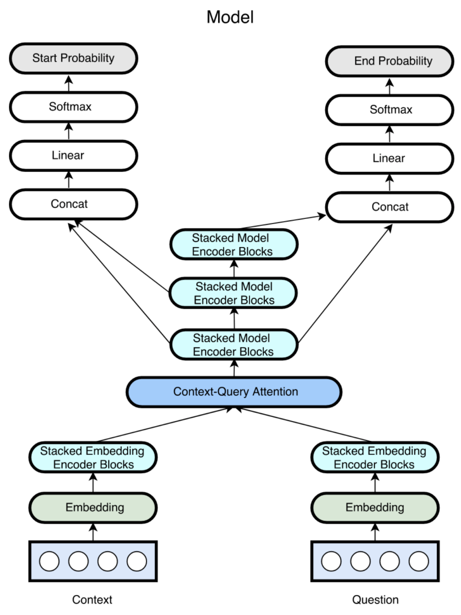
Model
Start Probability
End Probability
Softmax
Softmax
Linear
Linear
Concat
Concat
Stacked Model Encoder Blocks
Stacked Model Encoder Blocks
Stacked Model Encoder Blocks
Context-Query Attention
Stacked Embedding Encoder Blocks
Stacked Embedding Encoder Blocks
Embedding
Embedding
Context
Question

Figure 1: QANet Model as described in Yu et al. (2018) [1]

In the embedding-encoder layer we implement the QANet CNN-based transformer encoder, wherein encoder blocks consist of a consecutive triple of convolution, self-attention, and feed-forward layers. Each of these layers is placed inside a residual block and preceded by a layer norm. We also implemented sinusoidal absolute position encoding from scratch which we add to the input at the beginning of each encoder block.

In the context-attention layer we use the existing starter BiDAF code as this is the same as what is used in QANet, where context-to-query and query-to-context attention are computed by multiplying the row and column normalized similarity matrix of each pair of context and query words against the encoded query and context, respectively.

In the model encoder layer, we use the same transformer encoder block with convolutions used in the embedding-encoder layer except for the number of convolutional layers and block repetitions. These details are explained in more depth in the experimental setup section below.

In the output layer we use an approach similar to that of BiDAF, predicting the probability of each position in the context being the start or the end of an answer span. [2]

4.2 Improving QANet

After completing our version of QANet, we began implementing different architectures on top of our existing model. We discuss those below.

[2]We verified our implementation by looking at an open source QANet implementation

4.2.1 Data Augmentation

Because the training time of our model is significantly shorter than BiDAF, we decided to train our model on more data to attempt to improve performance. QANet takes advantage of this as described in [1], using 2-gram scores to find the highest probability paraphrased answer for each of the questions.

We also incorporate data augmentation into our model, but use our own technique designed to account for instances when the answer is “lost in translation." There are numerous cases where meaning is lost when translating across different languages, meaning there are cases where after backtranslating the context a question will no longer be answerable. To account for this, we implement a simple but stringent translation model: we translate the context and the answer separately, and only keep the (context, question, and answer) triple if the answer is an exact substring in the context paragraph. We therefore ensure that the answer within the context paragraph is not lost in the surrounding meaning through the process of backtranslation.[3]

4.2.2 Bidirectional Output Layers

As mentioned previously, we found the architecture for conditional output layers described in Kim and Wolff [4] to be a promising methodology for improving on the independence central to QANet's calculation of start and end probabilities in its output layer. As a result, we first implemented “Conditional output layer (2)" as described in Kim and Wolff [4], and then additionally extended this with two implementations of our own which build on this architecture with the following questions in mind: what is the relative importance of independent and conditional start/end probabilities; what value can be gained from conditioning both the end probabilities on the start probabilities, and the start probabilities on the end probabilities. Note that we avoid cyclic conditionality by using independent probabilities for conditions, not conditional probabilities.

Averaged Forward-Backward Output Layer

Our first attempt at improving upon the schema described in Kim and Wolff. [4] calculates the conditional start and end probabilities and averages them with the independent start and end probabilities, respectively. We begin by computing logits based on conditional “forward" (end conditioned on start) and “backward" (start conditioned on end) probabilities and then compute the start and end probabilities:

$$\begin{aligned}
S &= W_{0,S}[M_0; M_1] & E &= W_{0,E}[M_0; M_2] \\
A &= W_{1,S}(S \odot [M_0; M_1]) & X &= W_{1,E}(E \odot [M_0; M_2]) \\
B &= \text{ReLU}(W_{2,S}[M_0; M_2]) & Y &= \text{ReLU}(W_{2,E}[M_0; M_1]) \\
C &= W_{3,S}[A; B] & Z &= W_{3,E}[X; Y] \\
p_{\text{start}} &= \text{softmax}(\frac{S+Z}{2}) & p_{\text{end}} &= \text{softmax}(\frac{E+C}{2})
\end{aligned}$$

where $W_{0,S/E}, W_{3,S/E} \in \mathbb{R}^{1\times 2h}$ and $W_{1,S/E}, W_{2,S/E} \in \mathbb{R}^{h\times 2h}$ (h is the size of the hidden dimension).

True Forward-Backward Output Layer

Similar to our prior scheme, we also designed a “True Forward-Backward" scheme, which uses only conditioned probabilities to dictate start and end positions as opposed to using a mean of independent and conditioned start and end probabilities.

The equations for our “Average" and “True" layers are nearly identical, except for the final step:

[3]We use BackTranslation by PyPI

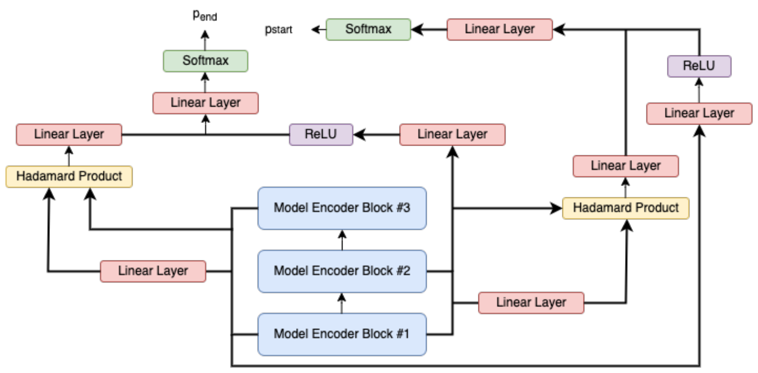

Figure 2: True Forward-Backward Output Layer

$$p_{\text{start}} = \text{softmax}(Z) \qquad\qquad p_{\text{end}} = \text{softmax}(C)$$

4.2.3 Ensembling

Finally, we implemented ensembling on top of our existing models, engaging in two different ensembling schemes: segment max and token max (these methods are described below). We ensemble over four different model architectures: baseline QANet; QANet+ [4]; QANet and Avg. For-Back; QANet and True For-Back. We use three different versions (differing in hyperparameters) of our baseline QANet model (as this achieved the highest F1 score amongst our individual models), and one version of the three other model architectures for a total of six models for ensembling.

Segment Ensembling

For each question and answer pair, segment max ensembling compares answers from each model against one other, and decides the "best" answer for each question by selecting the most frequent response on behalf of the models, breaking ties by using the model with the highest F1 score. We find this ensembling method productive as it combines choosing the most likely answer based on the possible answers while also preserving true answers (i.e. the chosen answer will be an answer generated by one of the models).

Token Ensembling

For each question and answer pair, token max ensembling iterates over each position in the answer and selects the "best" word for that position based on the most frequently appearing word at that position in the possible answers. We employ this method as it enables creating answers which can express greater granular attention to different possible answers.

5 Experiments

5.1 Data

We used the SQuAD (Stanford Question Answering Database) 2.0 dataset, created by Rajpurkar et al. (2018) [5], which serves as a particularly valuable annotated dataset because it's context, query, answer triples were created by humans through crowd-sourcing, making them more reflective of natural human language and interaction, and because it contains a subset of 50,000 unanswerable questions. The model will be trained on a predefined subset of SQuAD 2.0 and evaluated on the hidden test set, where the goal of the model is to be able to accurately answer all answerable test question and abstain from answering any unanswerable ones. As described in our section on Data Augmentation, we also created two new datasets, which we refer to as "SQuAD 2.0 Augmented v1" and "SQuAD 2.0 Augmented v2", which respectively introduce 8k and 7k additional QA pairs.

5.2 Evaluation method

In order to evaluate both our baseline and improved models, we use F1 score and EM score. F1 score, a measure of accuracy, is the mean of the precision and recall of the model's answer (i.e. $\frac{2 \times \text{prediction} \times \text{recall}}{\text{precision} + \text{recall}}$). Precision describes the percent of the predicted answer that is present in the ground-truth answer, and recall describes the percent of the ground-truth answer that was included in the predicted answer. EM score is a binary measure of whether or not the predicted output perfectly matches the ground truth answer. Since the F1 score is a more flexible measure of accuracy, it is considered more indicative of performance. We also evaluate our models based on training times. We compare the number of training steps and the amount of time it took the models to reach an approximate F1 score.

5.3 Experimental details

For our two baseline BiDAF models, we employ a constant learning rate of 0.5, a hidden size of 100, and a batch size of 64. For all of our QANet models, we use a constant learning rate of 0.001. Our regular QANet models use a hidden size of 128 and batch size of 64, although our models labled "Double" have hidden size of 256 and batch size of 48. We apply early stopping after we see F1 scores plateauing and train overfitting, so training times vary between our different models (these are listed in Table 1). We use a dropout rate of 0.1 between layers and during every other convolutional residual block in the encoder, and a dropout rate of 0.05 on character embeddings.

In our embedding encoder blocks, we use one iteration of the transformer block with 4 convolutional residual blocks. We use a kernel size of 7, filter size of 128, and group number of 128 (to make the convolutions depthwise separable, per [1]). In the model encoder blocks, we use seven iterations of the transformer block with 2 convolutional residual blocks for each of the three model encoders; we use a kernel size of 5 for these blocks.

We use the Adam optimizer with $\beta_1 = 0.8$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$, and L2 weight decay of $\lambda = 3 * 10^{-7}$ for our QANet models (BiDAF models use the Adadelta optimizer). We implemented our code in Pytorch and ran our implementations on a Tesla V100 GPU. We also use exponential moving average on all model parameters with decay of 0.9999.

5.4 Results

Model	F1	EM	AvNA	Training Time
BiDAF	60.99	57.79	67.5	2.5 hrs
BiDAF w/ Character Embed	64.12	60.85	70.48	2.75 hrs
QANet	**65.47**	**61.55**	72.51	2.75 hrs
QANet (Double) w/ Data Aug. v1	64.93	61.1	72.44	4.3 hrs
QANet (Double) w/ Data Aug. v2 (early stop)	58.81	55.22	66.48	1.75 hrs
QANet+	63.37	59.72	70.91	1.85 hrs
QANet w/ Avg. Forward-Backward Cond.	62.55	58.78	70.43	2.8 hrs
QANet w/ True Forward-Backward Cond.	64.51	60.8	71.8	2.6 hrs
Segment Max Ensemble (6 Models)	**67.712**	**64.544**	N/A	N/A
Token Max Ensemble (6 Models)	67.327	63.636	N/A	N/A
Test Leaderboard (Seg. Max 6 Models)	**63.998**	**60.828**	N/A	N/A

Table 1: F1/EM scores from baseline, improved, and ensemble models (dev scores unless otherwise stated).

The improvement in F1 score from BiDAF to BiDAF with Character Embed to QANet shows the impact of improvements to the embedding and encoding layers. Furthermore, we saw large speedups in training, as demonstrated by the fact that we achieved a similar F1 score to BiDAF 2.5x faster when using QANet. The performance trend was to be expected, but the relative score increases between adding character level embeddings and swapping RNNs for transformers were not. This implies that additional encoding information leads to a larger increase in score than learning global interactions between words.

We found that training our Double QANet model with Data Augmentation v1 as opposed to SQuAD 2.0 increases F1 score by a point, but still achieves a score slightly worse than our vanilla QANet implementation. However, we also found through ablation studies that training the vanilla QANet implementation on Data Augmentation v1 resulted in significantly lower performance (7 F1 score) than on SQuAD 2.0. We found these results surprising, as it negates the correlation between a larger dataset and better performance which we found logical in our results from Double QANet. Even more surprisingly, we found that training Double QANet on Data Augmentation v2 achieved a significantly lower score than Data Augmentation v1. We therefore found that data augmentation yields inconclusive results from the perspective of increasing performance.

For our studies on conditional output layers, we found that averaging conditional and independent probabilities performs worse than our implementation of QANet+ [4], but that using only bidirectional conditional probabilities over the hybrid expressed in QANet+ demonstrates notable improvement. As is clear in Kim and Wolff [4], using conditional probabilities improves performance over independent probabilities. Therefore, it logically follows that using conditional probabilities in both the same direction as described in [4] and also the opposite direction improves performance.

Through ensembling we saw significant improvement over the baseline QANet model ($+2.242$ F1). We also found that segment max outperforms token max; we expected this result, as segment max preserves existing answers as substrings of contexts, but token max can occasionally create answers which will not be contained within the context, meaning they are necessarily incorrect.

6 Analysis

6.1 Data Augmentation

As can be seen in Table 1, we performed experiments with different sized augmented datasets to understand the value of extra data as the amount of it increases. Data Augmentation v1 contained 8,000 additional question-answer pairs and achieved an F1 score of 64.93, while Data Augmentation v2 contained 15,000 additional question-answer pairs and achieved an F1 score of only 58.77 (we applied early stopping after dev loss began to increase rapidly). We conjecture that this decrease is due to the noise of backtranslated data which is compounded by our proposed method's lack of emphasis on the immediate context in which the answer occurs. Below is an example of a paraphrased context and answer generated by our data augmentation method.

	Context	Answer
Original	"Of the third Dalai Lama, China Daily states that the "Ming dynasty showed him special favor by allowing him to pay tribute. In 1653, the Qing emperor granted an honorific title to the fifth Dalai Lama and then did the same for the fifth Panchen Lama in 1713."	pay tribute
Paraphrase	"The third Dalai Lama, China daily, says China daily that the "Ming dynasty showed him special favor, he allowed him to pay tribute. In 1653, the Qing-Kaiser gave the fifth Dalai Lama a title of honor facility for the fifth Panchen Lama 1713."	honor

Table 2: Comparison between original context and answer and paraphrased context and answer.

As can be seen in Table 2, the backtranslation accurately paraphrases the first half of the context, but produces a different answer than the original. Our proposed ability to account for changes in meaning and structure of the augmented data result in this example being classified as a valid (context, question, answer) triple because of the usage of "honor" in the second half of the context. These errors became more significant as the proportion of augmented data in the overall dataset grew, negatively influencing the performance of our model. While the quality of augmented data likely could benefit from incorporating information on whether or not a question is answerable following backtranslation of the context, conditioning on the existence of a paraphrased answer within the context without looking at the context is evidently insufficient for creating consistently good performance on unseen questions and contexts.

6.2 Forward-Backward Conditionality

As demonstrated in our Results section, we found that True Forward-Backward conditionality outperforms QANet+, which in turn outperforms Average FB. At a high level, we attribute this to the fact that conditional and independent probabilities should not be weighted equally when deciding answer spans, as by using conditional probabilities for both the start and the end, we are able to establish a bidirectional relationship between both ends of the answer span. While True F-B conditionality shows signs of model improvement, the output layer appears to be an under-optimized area that should be explored going forward. With regards to why we see improved performance when using bidirectional conditional probabilities, we examine a sample answer that is predicted correctly by True FB and incorrectly by QANet+:

True Answer: electromagnetic force

QANet+ Answer: The electromagnetic force

Here, we see that QANet+ incorrectly predicts an overly long answer. More specifically, note that both answers have the same end token, which we expect as both QANet+ and True FB have end probabilities conditioned on start probabilities. However, note that QANet+ has an incorrect start token, which we can attribute to the fact that start probabilities are computed independently in QANet+. As a result, bidirectional conditionality enables additional output constraints which can lead to improved performance over unidirectional conditionality.

6.3 Ensembling

As seen in Table 1, segment max outperforms token max because it ensures the produced answer is a substring of the context. Token max is a greedy algorithm that for each token within the candidate answers selects the most commonly used token. However, since token max in no way considers context when choosing a token at a specific index, if just one token happens to be swapped outside of the most common answer, it is possible that the produced answer is no longer a substring of the context, which means it could not possibly be correct. Because EM score is a measure of perfect accuracy, we would expect the EM scores of the token max ensemble model to be more greatly affected than its F1, which we see more reflected in the dev leaderboard results. The token max ensemble scheme has F1 and EM score decreases of 0.385 and 0.908, respectively, against the segment max ensemble. Segment max, however, simply takes the maximum response across all of the models, so it does not suffer from the same shortcomings.

7 Conclusion

In this paper, we implement QANet and propose multiple extensions and improvements on that architecture, including a novel usage of ideas regarding conditional output layers proposed in Kim and Wolff [4]. We find that leveraging bidirectional or “alternating" conditional probabilities when deciding answer spans over unidirectional conditionality improves dev performance by 1.14 F1 and 1.08 EM. We also experiment with a unique data augmentation scheme based on the method proposed in Yu et al. [1], which yields inconclusive results. We finally implement alternate ensembling schemes which both show improvement over single-model performance, including a dev performance improvement of 2.242 F1 and 2.994 EM by our best ensembled model over our original model.

Our work is limited by the inconclusive results of data augmentation, as well as by the fact that our “AlterNet" model (i.e. QANet + True Forward-Backward output layer) performed about 1 F1 below our best vanilla QANet model on the dev set. At the same time, we learned that conditional probabilities are theoretically more important for span creation than independent probabilities (as suggested by the performance of Avg. Forward-Backward vs. True Forward-Backward). We also learned that data augmentation has possible benefits for larger models, but further research would be necessary to concretely understand this trend.

Moving forward, we think large improvements in performance could be achieved through further exploration of bidirectional conditionality in the output layer. Similar to the notion of generating greater global knowledge through coattention as introduced in [3], we find bidirectional conditional probabilities effective at simultaneously deriving contextual dependence between start and end tokens. Future work could extend our work on bidirectional conditionality to yield better task-specific results.

References

[1] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. Qanet: Combining local convolution with global self-attention for reading comprehension. In *International Conference on Learning Representations (ICLR)*, 2018.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *CoRR, abs/1706.03762*, 2017.

[3] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hananneh Hajishirzi. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations (ICLR)*, 2016.

[4] Moo Kim and Christopher Wolff. Qanet+: Improving qanet for question answering. In *CS224N Default Final Project*, 2020.

[5] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *arXiv preprint arXiv:1806.03822*, 2018.

[6] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. 2017.

[7] Jeffrey Pennington, Richard Socher, and Chris Manning. https://nlp.stanford.edu/pubs/glove.pdf. 2014.