Semi-Supervised Question Answering: Generative Augmentation in SQuAD2.0

Alex Lu
Department of Bioengineering
Stanford University
Stanford, CA 94305
alu2@stanford.edu

Abstract

The extractive question-answering task has been extensively investigated over the last years, especially with the advent of PCE methods such as BERT and ELMo. However, many current approaches are heavily dependent on large volumes of hand-labeled training data. Here, we investigate semi-supervised learning for SQuAD2.0 to better understand model dependence on training data, exploring multiple data augmentation methods that ameliorate effects of data restriction. We show that model performance is determined both by the underlying volume of data, as well as its quality.

1 Introduction

Question answering is a fundamental problem in natural language processing. The extractive question-answering formulation, in particular, has been extensively investigated over the last years. Given a context paragraph and a question, this formulation requires selection of a contiguous span of context words to answer the question. Significant progress has been made towards advancing the state of the art in extractive question-answering, both in SQuAD1.1, as well as with the recent introduction of the `no answer` possibility in SQuAD2.0 [1], [2]. Especially with the advent of PCE methods such as BERT and ELMo, significant progress has been made towards improving scores in the SQuAD challenge [3] [14]. As of this writing, BERT-based models are increasingly close to human performance on SQuAD2.0, reaching an F1 of 89.147 on the leaderboard.

Without a doubt, these results are impressive. One significant limitation that greatly affects NLP systems, however, is the need for an extensive dataset containing questions with associated contexts and answers. The SQuAD1.1 and SQuAD2.0 datasets of 100,000 examples required significant crowdsourcing and manual parsing to generate. With this dependency in mind, it is important to consider the robustness of extractive question-answering models under conditions in which such supervised data might not be available.

Here, we investigate semi-supervised question-answering methods to address the SQuAD2.0 challenge. In particular, I investigate various methods to reduce the dependence of extractive question-answering models on the availability of training data. These approaches include model simplification, naive (non-neural) data augmentation, and neural data augmentation through question generation. These approaches are compared in development set performance; the top performing model is evaluated against the training set.

2 Related Work

A number of works have approached the semi-supervised learning problem in extractive question answering. Generally, such approaches require a combination of existing supervised data as well as a pool of unsupervised data. Initial approaches were inspired by Yang et al., who identified two unique approaches to semi-supervised question-answering [6]. First, Yang et al. describe a naive NER and POS-based heuristic to extract contiguous answer phrases from unlabeled context paragraphs. The immediate k words before and after this answer phrase are joined to form the question. While Yang et al. noticed mild improvements with this naive method, they progressed into designing an architecture that couples a question-generative model with a discriminative question-answering model. These Generative Domain-Adaptive Networks worked well in SQuAD1.1, though were ultimately unable to advance the state of the art. Notably, in dividing the training of generative and discriminative models into three phases, it becomes difficult to identify bottlenecks in allowing the model to truly develop understanding of the context and question.

Question generation methods have also been employed by Wang et al. 2017 [7]. In their work, a generative model encodes context and candidate answers using a BiLSTM, before using a decoder consisting of two LSTM cells guided by a pointer-softmax mechanism. Notably, the work of Wang et al. uses a copy mechanism that chooses between copying from the context and generating from model vocabulary [8], [9]. The integration of artificially generated questions with existing training data allowed improvement on SQuAD relative to their baseline model.

Perhaps the most stark example of semi-supervised learning is devised by Dhingra et al., who achieved greater than .50 in $F1$ by leveraging less than a thousand labeled examples coupled with a unique question generation mechanism [10]. In their work, they cleverly utilize the larger scale structure of documents by decomposing them into introductory sentences $\{\mathbf{q_i}\}$ and content paragraphs $\{\mathbf{p_j}\}$ with the understanding that many key phrases within content paragraphs also appear within introductory sentences. If there is an exact phrase match in a paragraph $\mathbf{p_j}$ and a sentence $\mathbf{q_i}$, $\mathbf{q_i}$ with the phrase masked out will be utilized as the question; the phrase will be the answer, and the context paragraph will be the context. Perhaps it is more appropriate to refer to this method of question generation as reductive, capable of generating relatively syntactically fluent questions. On the whole, the performance achieved by Dhingra et al. is truly remarkable.

A final example to consider is that of BERT from Devlin et al. 2018. Notably, the pre-training phase of BERT utilizes the masked language model task, which is inspired by the same Cloze task as in Dhingra et al., but with the increased generality seen in the naive model of Wang et al. Additionally, while Wang et al. and Dhingra et al. aggregate artificial and human-generated questions to train in "one shot", BERT uses a two-phase combination of pre-training on the MLM task and sentence completion, which potentially allows the model to thoroughly learn how to extract meaning from text. As the foundation for much of the current SQuAD2.0 leaderboard, BERT is also an impressive step towards reducing dependence on hand-labeled data and devising ways to encourage model understanding from unlabeled text.

3 Approach

The goal of preserving or even improving extractive-question answering performance in SQuAD2.0 can be framed as one of trying to maximize performance on a development or test set $L_{\text{dev, test}} = \{q^{(i)}, a^{(i)}, p^{(i)}\}_{i=1}^{N_{\text{dev, test}}}$ while restricting the amount of supervised training data, $L_{\text{train}} = \{q^{(i)}, a^{(i)}, p^{(i)}\}_{i=1}^{N_{\text{train}}}$. Towards this end, two primary question generation approaches were considered. Question-answer pairs were artificially generated from unlabeled data either through naive methods resembling those of Yang et al. and Dhingra et al., as well as through neural question generation methods as described in Yang et al. and Wang et al. Additionally, model complexity was investigated. Overall, we hoped to encourage robust learning of the question-answering task while reducing dependence on training data volume.

3.1 Naive Data Augmentation

Question generation methods consist of two components: (1) extracting candidate answers from given context, (2) generating a question associated with each answer.

We implemented two naive methods that extract answers through preliminary Part-Of-Speech tagging and Named Entity Recognition to label tokenized context, followed by extraction of contiguous phrases. The question is then the set of W words on either side of this phrase, excluding the phrase itself. The first naive method selects random noun-phrases only, referred to in Table 1 as

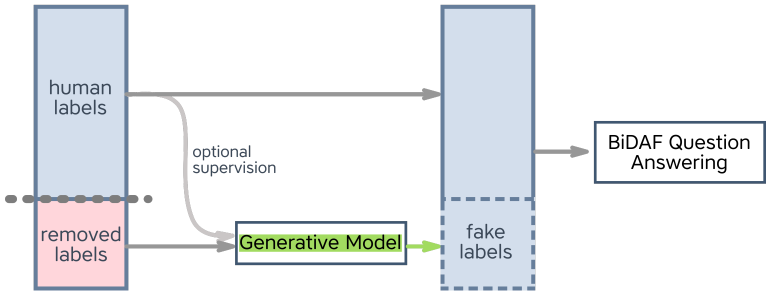

Figure 1: Semi-supervised Learning Pipeline

`random augmentation`. The second naive method performs more extensive labeling and selects candidate phrases based on their distribution in the SQuAD1.1 dataset [1]. Answers spans are then deliminated by the immediate left and right neighbors of the selected phrase in a dependency parse of the paragraph. NER, POS, and dependency tree functionalities were achieved using spaCy, `https://spacy.io`.

3.2 Neural Question Generation

A question generation model is trained using a subset of the SQuAD2.0 training set. The trained model then produces new question-answer-context triples over the unseen subset to augment supervised data for model training.

Embedding: The question generation model begins by using GloVe vectors to embed context, question, and answer indices [11]. This gives embedded vectors $\mathbf{c}_1, ..., \mathbf{c}_N \in \mathbb{R}^D$, $\mathbf{a}_1, ..., \mathbf{a}_M \in \mathbb{R}^D$, $\mathbf{q}_1, ..., \mathbf{q}_K \in \mathbb{R}^D$, where D is the embedding size, N gives the maximum context length, M gives the maximum answer length, and K gives the maximum question length.

Encoding: BiGRUs are used to encode the context and answer embedded vectors, producing $\mathbf{h}_i^c \in \mathbb{R}^H$, $\mathbf{h}_j^a \in \mathbb{R}^H$.

Answer2Context Attention: The Q2C attention mechanism from BiDAF is borrowed, instead deriving the attention distribution over context words hidden states relative to the provided answer hidden states [12]. As in BiDAF:

$$
\begin{aligned}
\mathbf{S}_{ij} &= \mathbf{w}_{\text{sim}}^T[\mathbf{h}_i^c; \mathbf{h}_j^a; \mathbf{h}_i^c \circ \mathbf{h}_j^a] \\
\bar{\mathbf{S}}_{i,:} &= \text{softmax}(\mathbf{S}_{i,:}) \quad \in \mathbb{R}^N \\
\bar{\bar{\mathbf{S}}}_{:,j} &= \text{softmax}(\mathbf{S}_{:,j}) \quad \in \mathbb{R}^M \\
\mathbf{S}' &= \bar{\mathbf{S}}\bar{\bar{\mathbf{S}}}^T \qquad \in \mathbb{R}^{N \times N} \\
\mathbf{b}_i &= \sum_{j=1}^{N} \mathbf{S}'_{i,j}\mathbf{c}_j \in \mathbb{R}^H \quad \forall i \in \{1, ..., N\}
\end{aligned}
$$

Decoding: The last hidden state of the context encoding is used to initialize the decoding during question generation. The embedded reference questions $\mathbf{q}_k$ are used to teacher-force the decoder, with $\bar{\mathbf{y}}^{(t)} = [\mathbf{y}^{(t)}; \mathbf{o}_{\text{prev}}] \in \mathbb{R}^{D+H}$. This is used to produce $\mathbf{h}_q^{(t)} = \text{GRUCell}(\bar{\mathbf{y}}^{(t)}, \text{dec-state}) \in \mathbb{R}^H$. This hidden state is re-mapped using a linear layer with no bias, producing $\mathbf{h}_q^{(t)\prime} = \mathbf{W}_{\text{dec-proj}} \in \mathbb{R}^H$, $\mathbf{W}_{\text{dec-proj}} \in \mathbb{R}^{H \times H}$.

Decode2Answer Attention: An attention distribution is generated over the answer hidden states using the decoder hidden state $e_j = (\mathbf{h}_j^a)^T \mathbf{W}_{\text{dec-proj}} \mathbf{h}_q^{(t)\prime} \in \mathbb{R}^K$ to produce a weighted combination of answer hidden states: $\mathbf{a_t} = [\mathbf{h}_1^a ... \mathbf{h}_K^a]\mathbf{e_t} \in \mathbb{R}^H$.

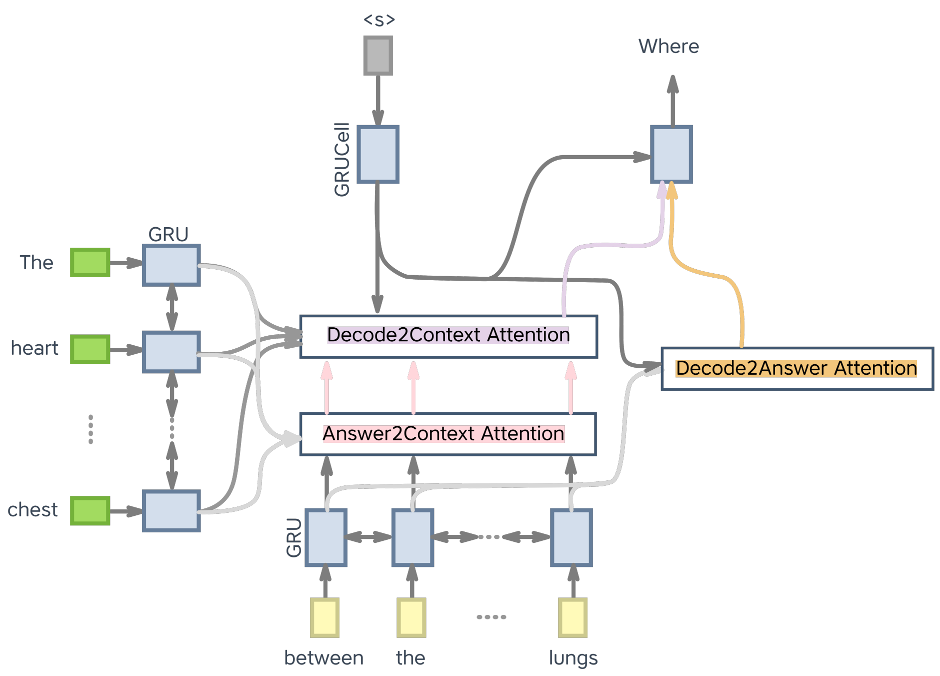

Figure 2: Neural Question Generation Architecture

Decode2Context Attention: An attention distribution is generated over the context hidden states using a combination of the Answer2Context attention and the decoder output.

$$\mathbf{v}_t = \mathbf{b}(\mathbf{W}_{\text{dec-proj}}\mathbf{h}_q^{(t)\prime})^T \quad \in \mathbb{R}^N$$
$$\mathbf{q}_t = [\mathbf{h}_1^c, ..., \mathbf{h}_N^c]\mathbf{v}_t \quad \in \mathbb{R}^H$$

Finally, we have the pre-output $\mathbf{S}_t = [\mathbf{h}_q^{(t)}; \mathbf{q}_t; \mathbf{h}_q^{(t)} \circ \mathbf{q}_t; \mathbf{a}_t] \in \mathbb{R}^{4H}$. The output of the decoding step is then given by: $\mathbf{o}_{\text{prev}} = \mathbf{W}_{\text{output-proj}}\mathbf{S}_t \in \mathbb{R}^H$.

The last step in decoding is given by a projection from the hidden state into the size of the vocabulary through a `tanh` activation, with a log softmax applied to give the log probabilities of the next output word. Beam-search is applied during evaluation and question generation modes to produce more fluent output. Drop-out is also applied throughout.

This architecture is visualized in Figure 2.

3.3 Models: Baseline versus Reduced Size

The baseline model is the provided implementation of bi-directional attention flow without character-level embedding. It contains an embedding layer, an encoding layer, bidirectional attention flow layer, a modeling layer, and output layer. In Table 1, this model is referred to as `baseline`.

In reducing the model size, we'd like to balance model expressivity while avoiding overfitting. With this in mind, the first reduced size model shares similar architecture with the baseline; however, the LSTM components of the encoding, modeling, and output layers are replaced by GRUs. LSTMs and GRUs are both used frequently in sequence modeling, but GRUs with identical hidden size have fewer parameters, and thus may better suit our purposes [15]. The reduced size model is referred to in Table 1 as `reduced`.

4 Experiments

A variety of combinations of training size restriction fractions and augmentation methods were used to explore dependency on labeled data. These experiments are documented in Table 1. The %augmentation column gives the total fraction of the training data that was substituted for augmentation.

4.1 Data

The provided default dataset for SQuAD2.0 is used. The available training data is partitioned to exclude 25%, 50%, with this excluded subset being used for question generation methods.

4.2 Evaluation Method

Overall model performance is ultimately evaluated using the development and test sets provided for our version of SQuAD2.0, using the AvNA, EM, and F1 metrics with different restrictions on the amount of hand-labeled training data. The question generation model, trained separately in maximizing negative log likelihood, is evaluated using BLEU.

5 Results

Table 1: Results

		Configuration				
Model	% supervised	% augmentation	Configuration	AvNA	EM	F1
`Baseline`	1.0	0.0	out of the box	67.00	57.00	60.35
`Baseline`	.50	0.0	out of the box	65.42	53.74	57.46
`Baseline`	.50	.50	random augmentation	67.84	56.61	60.20
`Baseline`	1.0	.25	random augmentation	69.01	59.38	62.71
`Baseline`	.75	.25	neural generation	47.86	32.3	37.06
`Reduced`	1.0	0.0		67.74	57.74	60.87
`Reduced`	.50	0.0		63.49	51.96	55.59
`Reduced`	.50	.50	random augmentation	66.95	56.16	59.85
`Reduced`	1.0	25	random augmentation	69.9	59.18	62.83
`Reduced`	.75	1.0	neural generation	47.86	32.45	37.15

5.1 Baselines

As seen in Table 1, the baseline model with 100% of training data available is able to achieve reasonable performance on SQuAD2.0. A second baseline is instantiated by reducing this training data, with no additional data augmentation. Both these results are displayed in the table.

5.2 Naive Augmentations

Different versions of naive augmentation were explored with different combinations of available training data. Notably, the 1:1 augmented baseline with most naive question generation mechanism was able to outperform the baseline without augmentation (even though most of the questions that are generated do not seem like real, fluent questions). Preliminary results further suggest high variability in model performance depending on which question-answer-context triples are ultimately sampled for training. A model trained on 50% of available data, with the remainder replaced by augmented data achieved EM of 54.235 and F1 of 57.741 on the test set, appearing close to the bottom of the leaderboard (: (). A resubmission with a model trained on all of available data supplemented with additional 25% augmented data climbed to rank 52 out of 78 on the test leaderboard with an EM of 58.698 and an F1 of 62.22. While this performance was not significant relative to the results of Dhingra et al., we can see that this one-shot aggregated approach of Cloze-like questions and random answers is apparently able to improve the baseline model's ability to learn from and understand the extractive question answering task.

While noun-phrase naive augmentation method was able to improve or ameliorate model performance under restrictive conditions, the augmentation seeking to match answer distribution with SQuAD1.1 was unable to achieve such results. Notably, the dependency tree methods used to extract the answer span often extracted answers that could be too long or too far apart, with associated context-based questions syntactically unrealistic or insufficiently distinguishable from other context regions. It remains to be explored whether better methods of extracting matched answer distributions will further improve model performance.

5.3 Question Generation

The question generation model was trained on 75% of the available training data, evaluated using BLEU on the dev set, and unexposed to the 25% partition of training data. After a hyperparameter search over learning rates, learning rate decay, hidden size, training time, and attention architecture, it seemed that the devised model could produce a peak score of roughly 22.00 corpus bleu. While the output questions are fairly fluent, it seems that a major flaw in the output questions is the relationship with the provided context and question (or lack thereof).

5.4 Baseline Performance with Aggregated Neural Augmentation

Looking at Table 1, we can see that questions produced through neural question generation actually worsen the performance of both `baseline` and `reduced` models. It's unlikely that this is due to limited fluency in generated questions – the naive augmentation methods produced far less fluent questions, and were able to improve model performance. In this situation, then, we can see that the impaired performance of the baseline is likely due to inability to generate questions that truly relate to the text. Perhaps a component of this shortcoming lies in the model's decoding mechanism, wherein the hidden state is projected into a vector the length of the corpus. Coupled with the greediness of beam search, sentence generation seems to favor statistically likely next words over rarer words with higher relationship to the context or question.

In fact, manual inspection of generated questions suggests that another issue is the inability to produce sufficient attention over the context words that directly correspond to the question, resulting in similar or identical questions given very different target answers. For instance, one context paragraph about Chopin successfully generated multiple instances of the question `What type of music did Chopin play?` for all three of the target answers `his earliest surviving musical`, `lessons`, `A-flat major`. Here, we can see that current model issues originate from struggling to learn interpretations of a single question corresponding to different answers. This could be improved by revising model architecture to incentivize unique, answer-driven questions.

6 Conclusion

On the whole, we have explored several data augmentation methods in attempts to reduce the reliance of extractive question-answering models on human-labeled data. We demonstrate a method for improving performance of the baseline model through a simple heuristic that allows learning to operate in a one-shot fashion, rather than requiring extended pre-training as in BERT [3]. We also construct a neural question generation model that successfully learns to produce fluent questions.

Though the use of neural questions impairs model performance, this behavior reinforces our understanding of several principles. First, the strong effects of a relatively small quantity of garbage neural questions show that the baseline model (and its reduced-size cousin) are able to extract understanding from the context-question pairs. This not only confirms the intuition underlying the ability of bidirectional attention flow to integrate significant context and question components, but also opens pathways towards utilizing adversarial methods in question-answer tasks. In the future, it will be worthwhile to explore the role of ensembling a question generation model with a question answering model during training to better provide online reinforcement and improve model robustness.

References

[1] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text. arXiv preprint. arxiv:1810.04805, 2016.

[2] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know What You Don't Know: Unanswerable Questions for SQuAD. arXiv preprint. ACL2018. arxiv:1806:03822. 2018.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805,2018.

[4] Xinya Du, Junru Shao, and Claire Cardie. Learning to Ask: Neural Question Generation for Reading Comprehension. https://arxiv.org/pdf/1705.00106.pdf

[5] Xingdi Yuan, Tong Wang, Caglar Gulcehre, et al. Machine Comprehension by Text-to-Text Neural Question Generation. https://arxiv.org/pdf/1705.02012.pdf

[6] Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William W. Cohen. Semi-Supervised QA with Generative Domain-Adaptive Nets. arXiv preprint arXiv:1702.02206v2, 2017.

[7] Tong Wang, Xingdi Yuan, and Adam Trischler. A Joint Model for Question Answering and Question Generation. https://arxiv.org/pdf/1706.01450.pdf

[8] Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. Pointing the Unknown Words. https://arxiv.org/pdf/1603.08148.pdf

[9] Jiatao Gu, Zhengdong Lu, Hang Li, Victor O.K. Li. Incorporating Copy Mechanisms in Sequence-to-Sequence Learning. arXiv preprint. arxiv:1603.06393v3

[10] Bhuwan Dhingra, Danish Pruthi, Dheeraj Rajagopal. Simple and Effective Semi-Supervised Question Answering. http://aclweb.org/anthology/N18-2092

[11] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global Vectors for Word Representation. https://nlp.stanford.edu/pubs/glove.pdf

[12] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectionalattention flow for machine comprehension. arXiv preprint arXiv:1611.01603, 2016.

[13] 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. arXiv preprint arXiv:1804.09541, 2018.

[14] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. Technical report, OpenAI. 2018.

[15] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. 2014. https://arxiv.org/pdf/1412.3555v1.pdf