

# Globally Normalized Reader

Jonathan Raiman and John Miller

Baidu Silicon Valley Artificial Intelligence Lab

{jonathanraiman,millerjohn}@baidu.com

## Abstract

Rapid progress has been made towards question answering (QA) systems that can extract answers from text. Existing neural approaches make use of expensive bi-directional attention mechanisms or score all possible answer spans, limiting scalability. We propose instead to cast extractive QA as an iterative search problem: select the answer's sentence, start word, and end word. This representation reduces the space of each search step and allows computation to be conditionally allocated to promising search paths. We show that globally normalizing the decision process and back-propagating through beam search makes this representation viable and learning efficient. We empirically demonstrate the benefits of this approach using our model, Globally Normalized Reader (GNR), which achieves the second highest single model performance on the Stanford Question Answering Dataset (68.4 EM, 76.21 F1 dev) and is 24.7x faster than bi-attention-flow. We also introduce a data-augmentation method to produce semantically valid examples by aligning named entities to a knowledge base and swapping them with new entities of the same type. This method improves the performance of all models considered in this work and is of independent interest for a variety of NLP tasks.

## 1 Introduction

Question answering (QA) and information extraction systems have proven to be invaluable in wide variety of applications such as medical information collection on drugs and genes (Quirk and

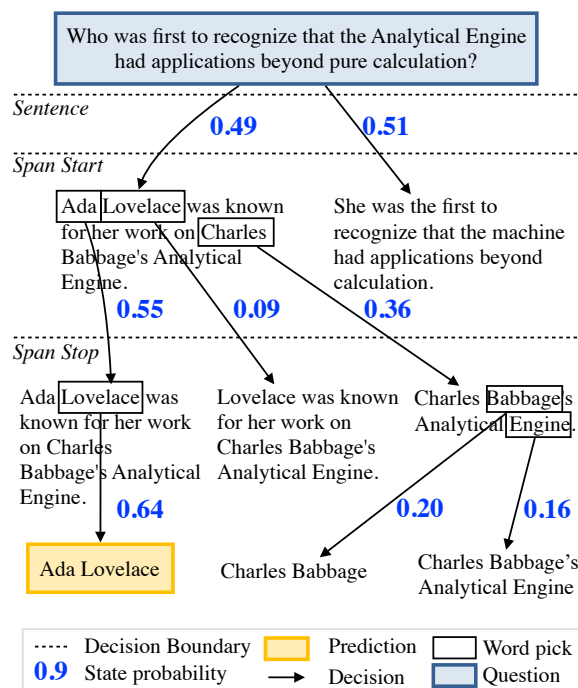


Figure 1: GNR answering a question. It first picks a sentence, then start word, then end word. Probabilities are global and normalized over the beam. Model initially picks the wrong sentence, but global normalization lets it recover. Final prediction's probability (0.64) exceeds sentence pick (0.49), whereas with local normalization each probability is upper bounded by the previous step.

Poon, 2016), large scale health impact studies (Althoff et al., 2016), or educational material development (Koedinger et al., 2015). Recent progress in neural-network based extractive question answering models are quickly closing the gap with human performance on several benchmark QA tasks such as SQuAD (Rajpurkar et al., 2016), MS MARCO (Nguyen et al., 2016), or NewsQA (Trischler et al., 2016a). However, current approaches to extractive question answering face several limitations:

1. Computation is allocated equally to the entire document, regardless of answer location, with no ability to ignore or focus computation on specific parts. This limits applicability to longer documents.
2. They rely extensively on expensive bi-directional attention mechanisms (Seo et al., 2016) or must rank all possible answer spans (Lee et al., 2016).
3. While data-augmentation for question answering have been proposed (Zhou et al., 2017), current approaches still do not provide training data that can *improve* the performance of existing systems.

In this paper we demonstrate a methodology for addressing these three limitations, and make the following claims:

1. Extractive Question Answering can be cast as a nested search process, where sentences provide a powerful document decomposition and an easy to learn search step. This factorization enables conditional computation to be allocated to sentences and spans likely to contain the right answer.
2. When cast as a search process, models without bi-directional attention mechanisms and without ranking all possible answer spans can achieve near state of the art results on extractive question answering.
3. Preserving narrative structure and explicitly incorporating type and question information into synthetic data generation is key to generating examples that actually improve the performance of question answering systems.

Our claims are supported by experiments on the SQuAD dataset where we show that the Globally Normalized Reader (GNR), a model that performs an iterative search process through a document (shown visually in Figure 1), and has computation conditionally allocated based on the search process, achieves near state of the art Exact Match (EM) and F1 scores without resorting to more expensive attention or ranking of all possible spans. Furthermore, we demonstrate that Type Swaps, a type-aware data augmentation strategy that aligns named entities with a knowledge base and swaps them out for new entities that share the same type,

improves the performance of all models on extractive question answering.

We structure the paper as follows: in Section 2 we introduce the task and our model. Section 3 describes our data-augmentation strategy. Section 4 introduces our experiments and results. In Section 5 we discuss our findings. In Section 6 we relate our work to existing approaches. Conclusions and directions for future work are given in Section 7.

## 2 Model

Given a document  $d$  and a question  $q$ , we pose extractive question answering as a search problem. First, we select the sentence, the first word of the span, and finally the last word of the span. An example of the output of the model is shown in Figure 1, and the network architecture is depicted in Figure 2.

More formally, let  $d_1, \dots, d_n$  denote each sentence in the document, and for each sentence  $d_i$ , let  $d_{i,1}, \dots, d_{i,m_i}$  denote the word vectors corresponding to the words in the sentence. Similarly, let  $q_1, \dots, q_\ell$  denote the word vectors corresponding to words in the question. An answer is a tuple  $a = (i^*, j^*, k^*)$  indicating the correct sentence  $i^*$ , start word in the sentence  $j^*$  and end word in the sentence  $k^*$ . Let  $\mathcal{A}(d)$  denote the set of valid answer tuples for document  $d$ . We now describe each stage of the model in turn.

### 2.1 Question Encoding

Each question is encoded by running a stack of bidirectional LSTM (Bi-LSTM) over each word in the question, producing hidden states  $(h_1^{\text{fwd}}, h_1^{\text{bwd}}), \dots, (h_\ell^{\text{fwd}}, h_\ell^{\text{bwd}})$  (Graves and Schmidhuber, 2005). Following Lee et al. (2016), these hidden states are used to compute a *passage-independent question embedding*,  $q^{\text{indep}}$ . Formally,

$$s_j = w_q^\top \text{MLP}([h_j^{\text{bwd}}; h_j^{\text{fwd}}]) \quad (1)$$

$$\alpha_j = \frac{\exp(s_j)}{\sum_{j'=1}^{\ell} \exp(s_{j'})} \quad (2)$$

$$q^{\text{indep}} = \sum_{j=1}^{\ell} \alpha_j [h_j^{\text{bwd}}; h_j^{\text{fwd}}], \quad (3)$$

where  $w_q$  is a trainable embedding vector, and MLP is a two-layer neural network with a Relu non-linearity. The question is represented by concatenating the final hidden states of the for-

ward and backward LSTMs and the passage-independent embedding,  $q = [h_1^{\text{bwd}}, h_\ell^{\text{fwd}}, q^{\text{indep}}]$ .

## 2.2 Question-Aware Document Encoding

Conditioned on the question vector, we compute a representation of each document word that is sensitive to both the surrounding context and the question. Specifically, each word in the document is represented as the concatenation of its word vector  $d_{i,j}$ , the question vector  $q$ , boolean features indicating if a word appears in the question or is repeated, and a *question-aligned embedding* from Lee et al. (2016). The question-aligned embedding  $q_{i,j}^{\text{align}}$  is given by

$$s_{i,j,k} = \text{MLP}(d_{i,j})^\top \text{MLP}(q_k) \quad (4)$$

$$\alpha_{i,j,k} = \frac{\exp(s_{i,j,k})}{\sum_{k'=1}^{\ell} \exp(s_{i,j,k'})} \quad (5)$$

$$q_{i,j}^{\text{align}} = \sum_{k=1}^{\ell} \alpha_{i,j,k} q_k. \quad (6)$$

The document is encoded by a separate stack of Bi-LSTMs, producing a sequence of hidden states  $(h_{1,1}^{\text{fwd}}, h_{1,1}^{\text{bwd}}), \dots, (h_{n,m_n}^{\text{fwd}}, h_{n,m_n}^{\text{bwd}})$ . The search procedure then operates on these hidden states.

## 2.3 Answer Selection

**Sentence selection.** The first phase of our search process picks the sentence that contains the answer span. Each sentence  $d_i$  is represented by the hidden state of the first and last word in the sentence for the backward and forward LSTM respectively,  $[h_{i,1}^{\text{bwd}}, h_{i,m_i}^{\text{fwd}}]$ , and is scored by passing this representation through a fully connected layer that outputs the unnormalized sentence score for sentence  $d_i$ , denoted  $\phi_{\text{sent}}(d_i)$ .

**Span start selection.** After selecting a sentence  $d_i$ , we pick the start of the answer span within the sentence. Each potential start word  $d_{i,j}$  is represented as its corresponding document encoding  $[h_{i,j}^{\text{fwd}}, h_{i,j}^{\text{bwd}}]$ , and is scored by passing this encoding through a fully connected layer that outputs the unnormalized start word score for word  $j$  in sentence  $i$ , denoted  $\phi_{\text{sw}}(d_{i,j})$ .

**Span end selection.** Conditioned on sentence  $d_i$  and starting word  $d_{i,j}$ , we select the end word from the remaining words in the sentence  $d_{i,j}, \dots, d_{i,m_i}$ . To do this, we run a Bi-LSTM over the remaining document hidden states

$(h_{i,j}^{\text{fwd}}, h_{i,j}^{\text{bwd}}), \dots, (h_{i,m_i}^{\text{fwd}}, h_{i,m_i}^{\text{bwd}})$  to produce representations  $(\tilde{h}_{i,j}^{\text{fwd}}, \tilde{h}_{i,j}^{\text{bwd}}), \dots, (\tilde{h}_{i,m_i}^{\text{fwd}}, \tilde{h}_{i,m_i}^{\text{bwd}})$ . Each end word  $d_{i,k}$  is then scored by passing  $[\tilde{h}_{i,j}^{\text{fwd}}, \tilde{h}_{i,j}^{\text{bwd}}]$  through a fully connected layer that outputs the unnormalized end word score for word  $k$  in sentence  $i$ , with start word  $j$ , denoted  $\phi_{\text{ew}}(d_{i,j:k})$ .

## 2.4 Global Normalization

The scores for each stage of our model can be normalized at the local or global level. Previous work demonstrated that locally-normalized models have a weak ability to correct mistakes made in previous decisions, while globally normalized models are strictly more expressive than locally normalized models (Andor et al., 2016; Zhou et al., 2015; Collins and Roark, 2004).

In a locally normalized model each decision is made conditional on the previous decision. The probability of some answer  $a = (i, j, k)$  is decomposed as

$$\mathbb{P}(a|d, q) = \mathbb{P}_{\text{sent}}(i|d, q) \cdot \mathbb{P}_{\text{sw}}(j|i, d, q) \cdot \mathbb{P}_{\text{ew}}(k|j, i, d, q). \quad (7)$$

Each sub-decision is locally normalized by applying a softmax to the relevant selection scores:

$$\mathbb{P}_{\text{sent}}(i|d, q) = \frac{\exp(\phi_{\text{sent}}(d_i))}{\sum_{x=1}^n \exp(\phi_{\text{sent}}(d_x))}, \quad (8)$$

$$\mathbb{P}_{\text{sw}}(j|i, d, q) = \frac{\exp(\phi_{\text{sw}}(d_{i,j}))}{\sum_{x=1}^{m_i} \exp(\phi_{\text{sw}}(d_{i,x}))}, \quad (9)$$

$$\mathbb{P}_{\text{ew}}(k|j, i, d, q) = \frac{\exp(\phi_{\text{ew}}(d_{i,j:k}))}{\sum_{x=j}^{m_i} \exp(\phi_{\text{ew}}(d_{i,j:x}))}. \quad (10)$$

To allow our model to recover from incorrect sentence or start word selections, we instead globally normalize the scores from each stage of our procedure. In a globally normalized model, we define

$$\text{score}(a, d, q) = \phi_{\text{sent}}(d_i) + \phi_{\text{sw}}(d_{i,j}) + \phi_{\text{ew}}(d_{i,j:k}). \quad (11)$$

Then, we model

$$\mathbb{P}(a | d, q) = \frac{\exp(\text{score}(a, d, q))}{Z}, \quad (12)$$

where  $Z$  is the partition function

$$Z = \sum_{a' \in \mathcal{A}(d)} \exp(\text{score}(a', d, q)). \quad (13)$$

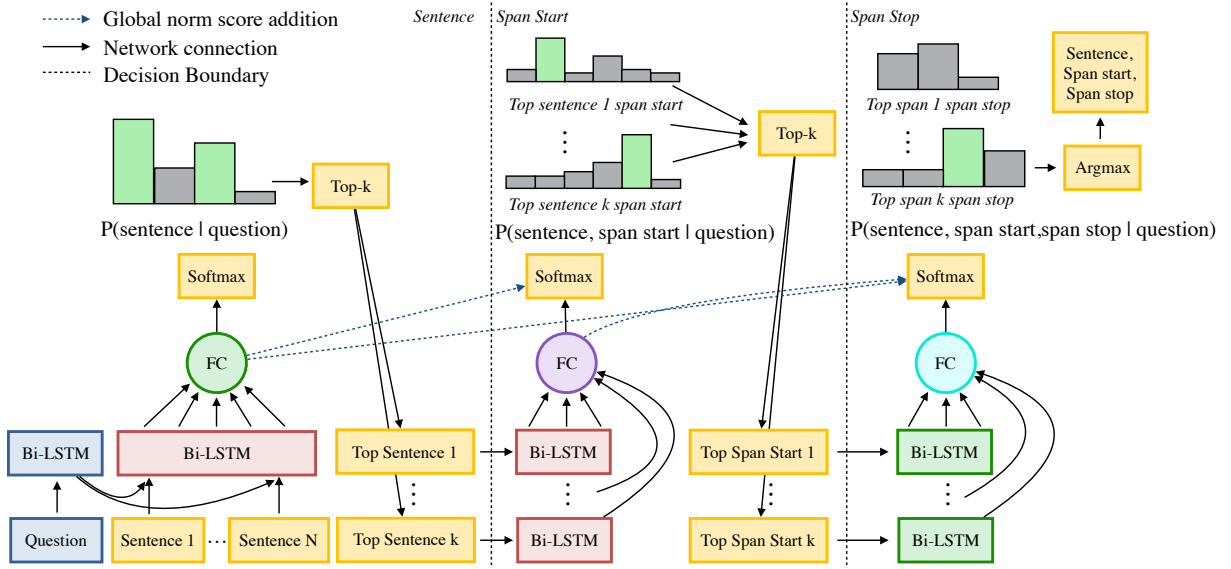


Figure 2: Globally Normalized Reader's search process. Same color Bi-LSTMs share weights.

In contrast to locally-normalized models, the model is normalized over all possible search paths instead of normalizing each step of search procedure. At inference time, the problem is to find

$$\arg \max_{a \in \mathcal{A}(d)} \mathbb{P}(a \mid d, q), \quad (14)$$

which can be approximately computed using beam search.

## 2.5 Objective and Training

We minimize the negative log-likelihood on the training set using stochastic gradient descent. For a single example  $(a, d, q)$ , the negative log-likelihood

$$-\text{score}(a, d, q) + \log Z \quad (15)$$

requires an expensive summation to compute  $\log Z$ . Instead, to ensure learning is efficient, we use beam search during training and early updates (Andor et al., 2016; Zhou et al., 2015; Collins and Roark, 2004). Concretely, we approximate  $Z$  by summing only over candidates on the final beam  $\mathcal{B}$ :

$$Z \approx \sum_{a' \in \mathcal{B}} \exp(\text{score}(a', d, q)). \quad (16)$$

At training time, if the gold sequence falls off the beam at step  $t$  during decoding, a stochastic gradient step is performed on the partial objective computed through step  $t$  and normalized over the beam at step  $t$ .

## 2.6 Implementation

Our best performing model uses a stack of 3 Bi-LSTMs for the question and document encodings, and a single Bi-LSTM for the end of span prediction. The hidden dimension of all recurrent layers is 200.

We use the 300 dimensional 8.4B token Common Crawl GloVe vectors (Pennington et al., 2014). Words missing from the Common Crawl vocabulary are set to zero. In our experiments, all architectures considered have sufficient capacity to overfit the training set. We regularize the models by fixing the word embeddings throughout training, dropping out the inputs of the Bi-LSTMs with probability 0.3 and the inputs to the fully-connected layers with probability 0.4 (Srivastava et al., 2014), and adding gaussian noise to the recurrent weights with  $\sigma = 10^{-6}$ . Our models are trained using Adam with a learning rate of 0.0005,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ ,  $\epsilon = 10^{-8}$  and a batch size of 32 (Kingma and Ba, 2014).

All our experiments are implemented in Tensorflow (Abadi et al., 2016), and we tokenize using Ciseau (Raiman, 2017). Despite performing beam-search during training, our model trains to convergence in under 4 hours through the use of efficient LSTM primitives in CuDNN (Chetlur et al., 2014) and batching our computation over examples and search beams. We release our code and augmented dataset.<sup>1</sup>

<sup>1</sup><https://github.com/baidu-research/GloballyNormalizedReader>

Our implementation of the GNR is 24.7 times faster at inference time than the official Bi-Directional Attention Flow implementation<sup>2</sup>. Specifically, on a machine running Ubuntu 14 with 40 Intel Xeon 2.6Ghz processors, 386GB of RAM, and a 12GB TitanX-Maxwell GPU, the GNR with beam size 32 and batch size 32 requires  $51.58 \pm 0.266$  seconds (mean  $\pm$  std)<sup>3</sup> to process the SQUAD validation set. By contrast, the Bi-Directional Attention Flow model with batch size 32 requires  $1260.23 \pm 17.26$  seconds. We attribute this speedup to avoiding expensive bi-directional attention mechanisms and making computation conditional on the search beams.

### 3 Type Swaps

**Question:** Who said in April 25, 2011 that the December 2012 fight would change from military to law enforcement?

**Answer:** Sheryl Sandberg

**Document (snippet):** ...Basic objectives of the Cabinet of Japan "war on terror", such as Bush administration targeting al Qaeda and building international counterterrorism alliances, remain in place. In April 25, 2011, Sheryl Sandberg, the General December 2012 Counsel of the Jeh Johnson Department of Defense ministry of education, stated that the military fight will be replaced by a law enforcement operation when speaking at Ain Shams University...

Oxford University

Figure 3: Type Swaps example. Replacements underlined with originals underneath.

In extractive question answering, the set of possible answer spans can be pruned by only keeping answers whose nature (person, object, place, date, etc.) agrees with the question type (Who, What, Where, When, etc.). While this heuristic helps human readers filter out irrelevant parts of a document when searching for information, no explicit supervision of this kind is present in the dataset. Despite this absence, the distribution question representations learned by our models appear to utilize this heuristic. The final hidden state of the

<sup>2</sup><https://github.com/allenai/bi-att-flow>

<sup>3</sup>All numbers are averaged over 5 runs.

question-encoding LSTMs naturally cluster based on question type (Table 1).

In other words, the task induces a question encoding that superficially respects type information. This property is a double-edged sword: it allows the model to easily weed out answers that are inapplicable, but also leads it astray by selecting a text span that shares the answer’s type but has the wrong underlying entity. A similar observation was made in the error analysis of (Weissenborn et al., 2017). We propose Type Swaps, an augmentation strategy that leverages this emergent behavior in order to improve the model’s ability to prune wrong answers, and make it more robust to surface form variation. This strategy has three steps:

1. Locate named entities in document and question.
2. Collect surface variation for each entity type:
 

$human \rightarrow \{\text{Ada Lovelace, Daniel Kahnemann, ...}\},$   
 $country \rightarrow \{\text{USA, France, ...}\}, \dots$
3. Generate new document-question-answer examples by swapping each named entity in an original triplet with a surface variant that shares the same type from the collection.

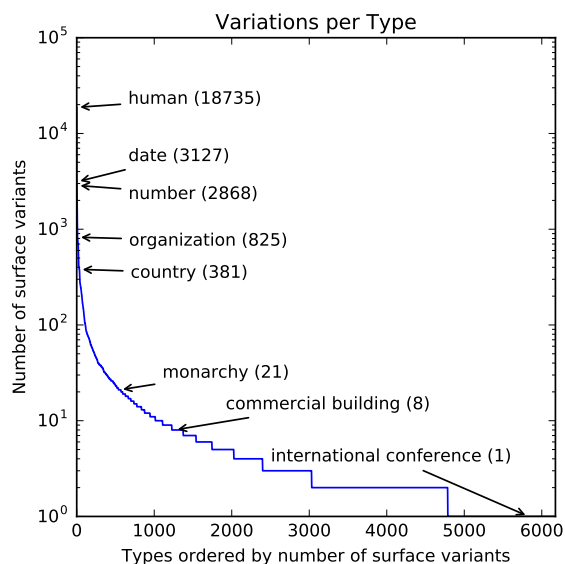


Figure 4: The majority of the surface variations occur for people, numbers, dates, and organizations.

Assigning types to named entities in natural language is an open problem, nonetheless when faced



Table 1: Top bigrams in K-means ( $K = 7$ ) clusters of question after Bi-LSTM. We observe emergent clustering according to question type: e.g. *Where*→ Cluster 7, *Who*→ Cluster 3. “What” granularity only observable with more clusters.

Cluster Size	1	2	3	4	5	6	7
	84789	42187	53061	130022	27549	16894	28377
Bigram	Bigram Occurences						
<i>what is</i>	3339	520	87	3736	20	8	138
<i>what did</i>	2463	3	3	112	1	0	1
<i>how many</i>	2	5095	1	1	0	0	0
<i>how much</i>	7	1102	0	12	0	0	0
<i>who was</i>	2	0	1934	0	0	0	1
<i>who did</i>	2	0	683	2	0	0	0
<i>what was</i>	2177	508	105	2034	71	31	92
<i>when did</i>	0	0	0	1	2772	0	0
<i>when was</i>	0	0	1	1	1876	0	0
<i>what year</i>	0	0	0	1	13	2690	0
<i>in what</i>	52	3	9	727	110	1827	518
<i>where did</i>	0	0	0	13	1	0	955
<i>where is</i>	0	1	0	11	0	0	665

with documents where we can safely assume that the majority of the entities will be contained in a large knowledge base (KB) such as Wikidata [Vrandečić and Krötzsch \(2014\)](#) we find that simple string matching techniques are sufficiently accurate. Specifically, we use a part of speech tagger ([Honnibal, 2017](#)) to extract nominal groups in the training data and string-match them with entities in Wikidata. Using this technique, we are able to extract 47,598 entities in SQuAD that fall under 6,380 Wikidata `instance of`<sup>4</sup> types. Additionally we assign “number types” (e.g. *year*, *day of the week*, *distance*, etc.) to nominal groups that contain dates, numbers, or quantities<sup>5</sup>. These extraction steps produce 84,632 unique surface variants (on average 16.93 per type) with the majority of the variation found in humans, numbers or organizations as visible in Figure 4.

With this method, we can generate  $2.92 \cdot 10^{369}$  unique documents (average of  $3.36 \cdot 10^{364}$  new documents for each original document). To ensure there is sufficient variation in the generated documents, we sample from this set and only keep variations where the question or answer is mutated. At each training epoch, we train on  $T$  Type Swap ex-

amples and the full original training data. An example output of the method is shown in Figure 3.

## 4 Results

We evaluate our model on the 100,000 example SQuAD dataset ([Rajpurkar et al., 2016](#)) and perform several ablations to evaluate the relative importance of the proposed methods.

### 4.1 Learning to Search

In our first experiment, we aim to quantify the importance of global normalization on the learning and search process. We use  $T = 10^4$  Type Swap samples and vary beam width  $B$  between 1 and 32 for a locally and globally normalized models and summarize the Exact-Match and F1 score of the model’s predicted answer and ground truth computed using the evaluation scripts from ([Rajpurkar et al., 2016](#)) (Table 3). We additionally report another metric, the *Sentence* score, which is a measure for how often the predicted answer came from the correct sentence. This metric provides a measure for where mistakes are made during prediction.

### 4.2 Type Swaps

In our second experiment, we evaluate the impact of the amount of augmented data on the performance of our model. In this experiment, we use

<sup>4</sup><https://www.wikidata.org/wiki/Property:P31>

<sup>5</sup>In our experiments we found that not including numerical variation in the generated examples led to an imbalanced dataset and lower final performance.

Table 2: Model comparison

Model	EM	F1
Human (Rajpurkar et al., 2016)	80.3	90.5
<i>Single model</i>		
Sliding Window (Rajpurkar et al., 2016)	13.3	20.2
Match-LSTM (Wang and Jiang, 2016)	64.1	73.9
DCN (Xiong et al., 2016)	65.4	75.6
Rasor (Lee et al., 2016)	66.4	74.9
Bi-Attention Flow (Seo et al., 2016)	67.7	77.3
R-Net(Wang et al., 2017)	<b>72.3</b>	<b>80.6</b>
Globally Normalized Reader w/o Type Swaps (Ours)	66.6	75.0
Globally Normalized Reader (Ours)	68.4	76.21

Table 3: Impact of Beam Width  $B$ 

Model	$B$	EM	F1	Sentence
Local, $T = 10^4$	1	65.7	74.8	<b>89.0</b>
	2	66.6	75.0	88.3
	10	66.7	75.0	88.6
	32	66.3	74.6	88.0
	64	66.6	75.0	88.8
	64	66.6	75.0	88.8
Global, $T = 10^4$	1	58.8	68.4	84.5
	2	64.3	73.0	86.8
	10	66.6	75.2	88.1
	32	<b>68.4</b>	<b>76.21</b>	88.4
	64	67.0	75.6	88.4
	64	67.0	75.6	88.4

the best beam sizes for each model ( $B = 10$  for local and  $B = 32$  for global) and vary the augmentation from  $T = 0$  (no augmentation) to  $T = 5 \cdot 10^4$ . The results of this experiment are summarized in (Table 4).

We observe that both models improve in performance with  $T > 0$  and performance degrades past  $T = 10^4$ . Moreover, data augmentation and global normalization are complementary. Combined, we obtain 1.6 EM and 2.0 F1 improvement over the locally normalized baseline.

We also verify that the effects of Type Swaps are not limited to our specific model by observing the impact of augmented data on the DCN+ (Xiong et al., 2016)<sup>6</sup>. We find that it strongly reduces generalization error, and helps improve F1, with potential further improvements coming by re-

<sup>6</sup> The DCN+ is the DCN with additional hyperparameter tuning by the same authors as submitted on the SQuAD leaderboard <https://rajpurkar.github.io/SQuAD-explorer/>.

Table 4: Impact of Augmentation Sample Size  $T$ .

Model	$T$	EM	F1	Sentence
Local	0	65.8	74.0	88.0
Local	$10^3$	66.3	74.6	88.9
Local	$10^4$	66.7	74.9	<b>89.0</b>
Local	$5 \cdot 10^4$	66.7	75.0	<b>89.0</b>
Local	$10^5$	66.2	74.5	88.6
Global	0	66.6	75.0	88.2
Global	$10^3$	66.9	75.0	88.1
Global	$10^4$	<b>68.4</b>	<b>76.21</b>	88.4
Global	$5 \cdot 10^4$	66.8	75.3	88.3
Global	$10^5$	66.1	74.3	86.9

Table 5: Impact of Type Swaps on the DCN+

$T$	Train F1	Dev F1
0	81.3	78.1
$5 \cdot 10^4$	72.5	<b>78.2</b>

ducing other forms of regularization (Table 5).

## 5 Discussion

In this section we will discuss the results presented in Section 4, and explain how they relate to our main claims.

### 5.1 Extractive Question Answering as a Search Problem

Sentences provide a natural and powerful document decomposition for search that can be easily learnt as a search step: for all the models and configurations considered, the *Sentence* score was

above 88% correct (Table 3)<sup>7</sup>. Thus, sentence selection is the easy part of the problem, and the model can allocate more computation (such as the end-word selection Bi-LSTM) to spans likely to contain the answer. This approach avoids wasteful work on unpromising spans and is important for further scaling these methods to long documents.

## 5.2 Global Normalization

The Globally Normalized Reader outperforms previous approaches and achieves the second highest EM behind (Wang et al., 2017), without using bi-directional attention and only scoring spans in its final beam. Increasing the beam width improves the results for both locally and globally normalized models (Table 3), suggesting search errors account for a significant portion of the performance difference between models. Models such as Lee et al. (2016) and Wang and Jiang (2016) overcome this difficulty by ranking all possible spans and thus never skipping a possible answer. Even with large beam sizes, the locally normalized model underperforms these approaches. However, by increasing model flexibility and performing search during training, the globally normalized model is able to recover from search errors and achieve much of the benefits of scoring all possible spans.

## 5.3 Type-Aware Data Augmentation

Type Swaps, our data augmentation strategy, offers a way to incorporate the nature of the question and the types of named entities in the answers into the learning process of our model and reduce sensitivity to surface variation. Existing neural-network approaches to extractive QA have so far ignored this information. Augmenting the dataset with additional type-sensitive synthetic examples improves performance by providing better coverage of different answer types. Growing the number of augmented samples used improves the performance of all models under study (Table 4-5). With  $T \in [10^4, 5 \cdot 10^4]$ , (EM, F1) improve from (65.8  $\rightarrow$  66.7, 74.0  $\rightarrow$  75.0) for locally normalized models, and (66.6  $\rightarrow$  68.4, 75.0  $\rightarrow$  76.21)

<sup>7</sup>The objective function difference explains the lower performance of globally versus locally normalized models on the *Sentence* score: local models must always assign the highest probability to the correct sentence, while global models only ensure the correct span has the highest probability. Thus global models do not need to enforce a high margin between the correct answer’s sentence score and others and are more likely to keep alternate sentences around.

for globally normalized models.

Past a certain amount of augmentation, we observe performance degradation. This suggests that despite efforts to closely mimic the original training set, there is a train-test mismatch or excess duplication in the generated examples.

Our experiments are conducted on two vastly different architectures and thus these benefits are expected to carry over to different models (Weissenborn et al., 2017; Seo et al., 2016; Wang et al., 2017), and perhaps more broadly in other natural language tasks that contain named entities and have limited supervised data.

## 6 Related Work

Our work is closely related to existing approaches in learning to search, extractive question answering, and data augmentation for NLP tasks.

**Learning to Search.** Several approaches to learning to search have been proposed for various NLP tasks and conditional computation. Most recently, Andor et al. (2016) and Zhou et al. (2015) demonstrated the effectiveness of globally normalized networks and training with beam search for part of speech tagging and transition-based dependency parsing, while Wiseman and Rush (2016) showed that these techniques could also be applied to sequence-to-sequence models in several application areas including machine translation. These works focus on parsing and sequence prediction tasks and have a fixed computation regardless of the search path, while we show that the same techniques can also be straightforwardly applied to question answering and extended to allow for conditional computation based on the search path.

Learning to search has also been used in context of modular neural networks with conditional computation in the work of Andreas et al. (2016) for image captioning. In their work reinforcement learning was used to learn how to turn on and off computation, while we find that conditional computation can be easily learnt with maximum likelihood and the help of early updates (Andor et al., 2016; Zhou et al., 2015; Collins and Roark, 2004) to guide the training process.

Our framework for conditional computation whereby the search space is pruned by a sequence of increasingly complex models is broadly reminiscent of the structured prediction cascades of (Weiss and Taskar, 2010). Trischler et al. (2016b)



also explored this approach in the context of question answering.

**Extractive Question Answering.** Since the introduction of the SQuAD dataset, numerous systems have achieved strong results. Seo et al. (2016); Wang et al. (2017) and Xiong et al. (2016) make use of a bi-directional attention mechanisms, whereas the GNR is more lightweight and achieves similar results without this type of attention mechanism. The document representation used by the GNR is very similar to Lee et al. (2016). However, both Lee et al. (2016) and Wang and Jiang (2016) must score all  $O(N^2)$  possible answer spans, making training and inference expensive. The GNR avoids this complexity by learning to search during training and outperforms both systems while scoring only  $O(B)$  spans. Weissenborn et al. (2017) is a locally normalized model that first predicts start and then end words of each span. Our experiments lead us to believe that further factorizing the problem and using global normalization along with our data augmentation would yield corresponding improvements.

**Data augmentation.** Several works use data augmentation to control the generalization error of deep learning models. Zhang and LeCun (2015) use a thesaurus to generate new training examples based on synonyms. Vijayaraghavan et al. (2016) employs a similar method, but uses Word2vec and cosine similarity to find similar words. Jia and Liang (2016) use a high-precision synchronous context-free grammar to generate new semantic parsing examples. Our data augmentation technique, Type Swaps, is unique in that it leverages an external knowledge-base to provide new examples that have more variation and finer-grained changes than methods that use only a thesaurus or Word2Vec, while also keeping the narrative and grammatical structure intact.

More recently Zhou et al. (2017) proposed a sequence-to-sequence model to generate diverse and realistic training question-answer pairs on SQuAD. Similar to their approach, our technique makes use of existing examples to produce new examples that are fluent, however we also are able to explicitly incorporate entity type information into the generation process and use the generated data to improve the performance of question answering models.

## 7 Conclusions and Future Work

In this work, we provide a methodology that overcomes several limitations of existing approaches to extractive question answering. In particular, our proposed model, the Globally Normalized Reader, reduces the computational complexity of previous models by casting the question answering as search and allocating more computation to promising answer spans. Empirically, we find that this approach, combined with global normalization and beam search during training, leads to near state of the art results. Furthermore, we find that a type-aware data augmentation strategy improves the performance of all models under study on the SQuAD dataset. The method is general, only requiring that the training data contains named entities from a large KB. We expect it to be applicable to other NLP tasks that would benefit from more training data.

As future work we plan to apply the GNR to other question answering datasets such as MS MARCO (Nguyen et al., 2016) or NewsQA (Trischler et al., 2016a), as well as investigate the applicability and benefits of Type Swaps to other tasks like named entity recognition, entity linking, machine translation, and summarization. Finally, we believe there a broad range of structured prediction problems (code generation, generative models for images, audio, or videos) where the size of original search space makes current techniques intractable, but if cast as learning-to-search problems with conditional computation, might be within reach.

## Acknowledgments

We would like to thank the anonymous reviewers for their valuable feedback. In addition, we thank Adam Coates, Carl Case, Andrew Gibiansky, and Szymon Sidor for thoughtful comments and fruitful discussion. We also thank James Bradbury and Bryan McCann for running Type Swap experiments on the DCN+.

## References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

- Tim Althoff, Ryen W White, and Eric Horvitz. 2016. Influence of pokémon go on physical activity: Study and implications. *Journal of Medical Internet Research*, 18(12).
- Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. *arXiv preprint arXiv:1603.06042*.
- Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. *arXiv preprint arXiv:1601.01705*.
- Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. 2014. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*.
- Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 111. Association for Computational Linguistics.
- Alex Graves and Jürgen Schmidhuber. 2005. Frame-wise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.
- Matthew Honnibal. 2017. Spacy. <https://github.com/explosion/spaCy>.
- Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. *arXiv preprint arXiv:1606.03622*.
- Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Kenneth R Koedinger, Sidney D’Mello, Elizabeth A McLaughlin, Zachary A Pardos, and Carolyn P Rosé. 2015. Data mining and education. *Wiley Interdisciplinary Reviews: Cognitive Science*, 6(4):333–353.
- Kenton Lee, Tom Kwiatkowski, Ankur Parikh, and Dipanjan Das. 2016. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436*.
- Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.
- Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.
- Chris Quirk and Hoifung Poon. 2016. Distant supervision for relation extraction beyond the sentence boundary. *arXiv preprint arXiv:1609.04873*.
- Jonathan Raiman. 2017. Ciseau. <https://github.com/jonathanraiman/ciseau>.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.
- Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.
- Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.
- Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016a. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*.
- Adam Trischler, Zheng Ye, Xingdi Yuan, Jing He, Phillip Bachman, and Kaheer Suleman. 2016b. A parallel-hierarchical model for machine comprehension on sparse data. *arXiv preprint arXiv:1603.08884*.
- Prashanth Vijayaraghavan, Ivan Sysoev, Soroush Vosoughi, and Deb Roy. 2016. Deepstance at semeval-2016 task 6: Detecting stance in tweets using character and word-level cnns. *arXiv preprint arXiv:1606.05694*.
- Denny Vrandečić and Markus Krötzsch. 2014. Wiki-data: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.
- Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*.
- Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.
- David Weiss and Benjamin Taskar. 2010. Structured prediction cascades. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 916–923.
- Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Fastqa: A simple and efficient neural architecture for question answering. *arXiv preprint arXiv:1703.04816*.
- Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. *arXiv preprint arXiv:1606.02960*.

- Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*.
- Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*.
- Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *ACL (1)*, pages 1213–1222.
- Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. *arXiv preprint arXiv:1704.01792*.