Generating Well-Formed Answers by Machine Reading with Stochastic Selector Networks

Bin Bi, Chen Wu, Ming Yan, Wei Wang, Jiangnan Xia, Chenliang Li
Alibaba Group
{b.bi, wuchen.wc, ym119608, hebian.ww, jiangnan.xjn, lcl193798}@alibaba-inc.com

Abstract

Question answering (QA) based on machine reading comprehension has been a recent surge in popularity, yet most work has focused on extractive methods. We instead address a more challenging QA problem of generating a well-formed answer by reading and summarizing the paragraph for a given question.

For the generative QA task, we introduce a new neural architecture, LatentQA, in which a novel stochastic selector network composes a well-formed answer with words selected from the question, the paragraph and the global vocabulary, based on a sequence of discrete latent variables. Bayesian inference for the latent variables is performed to train the LatentQA model. The experiments on public datasets of natural answer generation confirm the effectiveness of LatentQA in generating high-quality well-formed answers.

Introduction

Question answering (QA) is an essential problem in natural language understanding and a major milestone towards human-level machine intelligence. Machine reading comprehension, which enables machines to answer questions after reading documents, has become a popular and attractive solution to question answering in recent years (Rajpurkar et al. 2016; Rajpurkar, Jia, and Liang 2018; Nguyen et al. 2016). Existing techniques for machine reading comprehension fall primarily into the category of extractive methods, which extract a piece of text from a contextual paragraph as an answer to a given question. The extractive answer comprised of a few words is restricted to be an exact sub-span in the paragraph.

In real-world applications, however, a span of text is often insufficient to answer a question, such as How/Why questions that lead to long answers. Generating an abstractive answer is needed instead, which requires a QA system to summarize the content in the paragraph that is relevant to the question. Moreover, users prefer answers that can be read in a standalone fashion. Such well-formed answers not only address the questions, but also provide supporting information or explanation, so that users make sense of the answers without the need for perfect context.

Generating well-formed answers can be beneficial to a variety of QA applications. For example, digital agents, such as Siri, Google Assistant, Cortana and Alexa, are designed to respond to a question by reading out the well-formed answer in natural language. In this scenario, the answers need to be standalone and self-contained, as users are not expected to understand full context.

In this paper, we present a new neural architecture, *LatentQA*, which generates well-formed answers to given questions by reading contextual paragraphs. Unlike existing answer generation models which add a decoder on top of extractive models, LatentQA neither relies on extraction results, nor needs labels of answer spans for training. Instead, LatentQA resorts to a novel *stochastic selector network* that selects words to form a final answer directly from the modeled relationship between the question and the paragraph. In the stochastic selector network, a sequence of discrete latent variables is introduced to indicate which source to look at to produce every answer word. A word in a well-formed answer comes from one of three sources: the question, the paragraph or the global vocabulary. To train LatentQA, we perform Bayesian inference for the latent variables, and derive the posterior probabilities.

With the stochastic mechanism, LatentQA is able to model the ambiguity inherent in questions and paragraphs, and to generate final answers based on the interpretations. Furthermore, LatentQA is more robust against overfitting than deterministic models, by marginalizing over the latent variables. We conduct experiments on two public datasets of natural answer generation, MARCO and DuReader. The empirical evaluation confirms the effectiveness of LatentQA in generating high-quality answers.

Related Work

Machine Reading Comprehension. Machine reading has made rapid progress in recent years. such as SQuAD (Rajpurkar et al. 2016; Rajpurkar, Jia, and Liang 2018). The majority of studies treat reading comprehension as answer span extraction from a given paragraph, which is normally achieved by predicting the start and end position of an answer. Seo et al. (2017) proposed BiDAF that represents context at different levels of granularity and uses the bi-directional attention flow mechanism for answer extraction. SLQA (Wang, Yan, and Wu 2018) improves answer quality with a hierarchical attention fusion network in which attention and fusion are conducted horizontally and vertically across layers between the question and the paragraph. Recently, we see emerging BERT-based models (Devlin et al. 2018) which are proven effective for reading comprehension. Multi-paragraph reading comprehension has also attracted interest from the academic (Yan et al. 2019) and industrial community (He et al. 2018).

Sequence-to-sequence QA. The sequence-to-sequence architecture has been broadly used in a variety of QA tasks without reading contextual paragarphs. GenQA (Yin et al. 2016) combines knowledge retrieval and sequence-to-sequence learning to produce fluent answers, but it only deals with simple questions containing one single fact. COREQA (He et al. 2017) extends it with a copy mechanism, and can answer an information inquired question (i.e., a factual question containing one or more topic entities). In contrast, Fu and Feng (2018) introduced a new attention mechanism that explores heterogeneous memory for answer sentence generation. The new attention encourages the decoder to actively interact with the memory in the memory-augmented encoder-decoder framework. Moreover, Tao et al. (2018) proposed a multi-head attention mechanism to capture multiple semantic aspects of a given query and generate an informative response in a dialogue system.

Natural Answer Generation. There have been several attempts at using machine reading to generate natural answers. Tan et al. (2018) took a generative approach where they added a decoder on top of their extractive model to leverage the extracted evidence for answer synthesis. However, this model still relies heavily on the extraction to perform the generation and thus needs to have start and end labels (a span) for every QA pair. Mitra (2017) proposed a seq2seq-based model that learns alignment between a question and passage words to produce rich question-aware passage representation by which it directly decodes an answer. Gao et al. (2019) focused on product-aware answer generation based on large-scale unlabeled e-commerce reviews and product attributes. Furthermore, natural answer generation can be reformulated as query-focused summarization (QFS) which is addressed by Nema et al. (2017) as well as Hasselqvist, Helmertz, and Kågebäck (2017). Recently, the Masque model from Nishida et al. (2019) explored the idea of copying words from questions to answers with a mixture of multiple distributions. Our model differs from Masque in that LatentQA models inherent uncertainty with a stochastic mechanism and integrates a dedicated selector network to exploit all three information sources for well-formed answer generation. Pre-trained contextualized representations, such as ELMo (Peters et al. 2018) used in Masque, can be readily plugged into the stochastic selector networks for further improvement.

Paragraph	Bake sirloin steaks in the oven at 425 degrees Fahrenheit for 30 minutes until they are cooked to your desired taste. Baking sirloin steaks decreases the moisture available in the steaks. The oven tends to dry the meat out if you do not take the time to marinate appropriately.
Question	How long to cook sirloin steak?
Well-formed Answer	It takes 30 minutes to cook sirloin steak in the oven at 425 degrees Fahrenheit.

Table 1: A sample well-formed answer with words in green from the vocabulary, words in red from the paragraph, and words in blue from the question.

Well-formed Answer Generation

Well-formed answer generation is a question answering paradigm where a QA model is expected to answer a given question in a way that is understood without perfect context. More formally, let (q, a, p) denote an instance from a QA dataset of N instances, where q denotes a question, a denotes an answer, and p denotes a paragraph. Well-formed answer generation aims to produce an abstractive answer a to a given question q based on the content in paragraph p. Different from extractive QA, generated answer a does not have to be a sub-span in paragraph p. Instead, answer a is supposed to be formed in natural language, and to make sense without the context of either question q or paragraph p.

LatentQA

In composing a well-formed answer a, our QA model, LatentQA, recurrently selects words at the decoding stage. Traditional QFS and answer generation models select words from either vocabulary v alone (Tan et al. 2018; Nema et al. 2017) or a combination of vocabulary v and paragraph p (Mitra 2017). However, when it comes to generating well-formed answers, the two sources v and p are often insufficient to provide the answers with proper context.

In contrast, LatentQA employs a novel stochastic selector network for answer composition, which allows answer words to come from three different sources: question q, paragraph p, and vocabulary v. Table 1 shows a specific example of a well-formed answer generated by selecting words from the three sources. An overview of the architecture of LatentQA is depicted in Figure 1.

Sequence-to-sequence model

LatentQA is built upon an extension of the sequence-to-sequence model (Bahdanau, Cho, and Bengio 2015; Nallapati et al. 2016; See, Liu, and Manning 2017). The words of question q and paragraph p are fed one-by-one into two different encoders, respectively. Each of the two encoders,

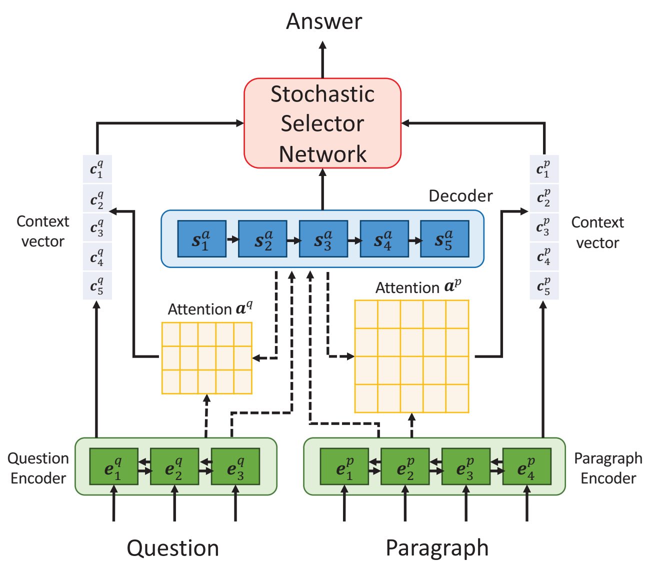

Figure 1: An overview of the architecture of LatentQA (best viewed in color). A question and a paragraph both go through an extension of the sequence-to-sequence model. The outcomes are then fed into the stochastic selector network to generate a well-formed answer.

which are both bidirectional LSTMs, produces a sequence of encoder hidden states ($\mathbf{E}^q$ for question q, and $\mathbf{E}^p$ for paragraph p). In each timestep t, the decoder, which is a unidirectional LSTM, takes an answer word as input, and outputs a decoder state $\mathbf{s}_t^a$.

We calculate attention distributions $\mathbf{a}_t^q$ and $\mathbf{a}_t^p$ on the question and the paragraph, respectively, as in (Bahdanau, Cho, and Bengio 2015):

$$\mathbf{a}_t^q = \text{softmax}(\mathbf{g}^{q\intercal}\tanh(\mathbf{W}^q\mathbf{E}^q + \mathbf{U}^q\mathbf{s}_t^r + \mathbf{b}^q)), \quad (1)$$

$$\mathbf{a}_t^p = \text{softmax}(\mathbf{g}^{p\intercal}\tanh(\mathbf{W}^p\mathbf{E}^p + \mathbf{U}^p\mathbf{s}_t^r + \mathbf{V}^p\mathbf{c}^q + \mathbf{b}^p)), \quad (2)$$

where $\mathbf{g}^q$, $\mathbf{W}^q$, $\mathbf{U}^q$, $\mathbf{b}^q$, $\mathbf{g}^p$, $\mathbf{W}^p$, $\mathbf{U}^p$ and $\mathbf{b}^p$ are learnable parameters. The attention distributions can be viewed as probability distributions over source words, which tells the decoder where to look to generate the next word. The coverage mechanism is added to the attentions to avoid generating repetitive text (See, Liu, and Manning 2017). In Equation 2, we introduce $\mathbf{c}^q$, a context vector for the question, to make the paragraph attention aware of the question context. $\mathbf{c}^q$ for the question and $\mathbf{c}^p$ for the paragraph are calculated as follows:

$$\mathbf{c}_t^q = \sum_i a_{ti}^q \cdot \mathbf{e}_i^q, \qquad \mathbf{c}_t^p = \sum_i a_{ti}^p \cdot \mathbf{e}_i^p, \quad (3)$$

where $\mathbf{e}_i^q$ and $\mathbf{e}_i^p$ are an encoder hidden state for question q and paragraph p, respectively. The context vectors ($\mathbf{c}_t^q$ and $\mathbf{c}_t^p$) together with the attention distributions ($\mathbf{a}_t^q$ and $\mathbf{a}_t^p$) and the decoder state ($\mathbf{s}_t^a$) will be used downstream to determine the next word in composing a final answer.

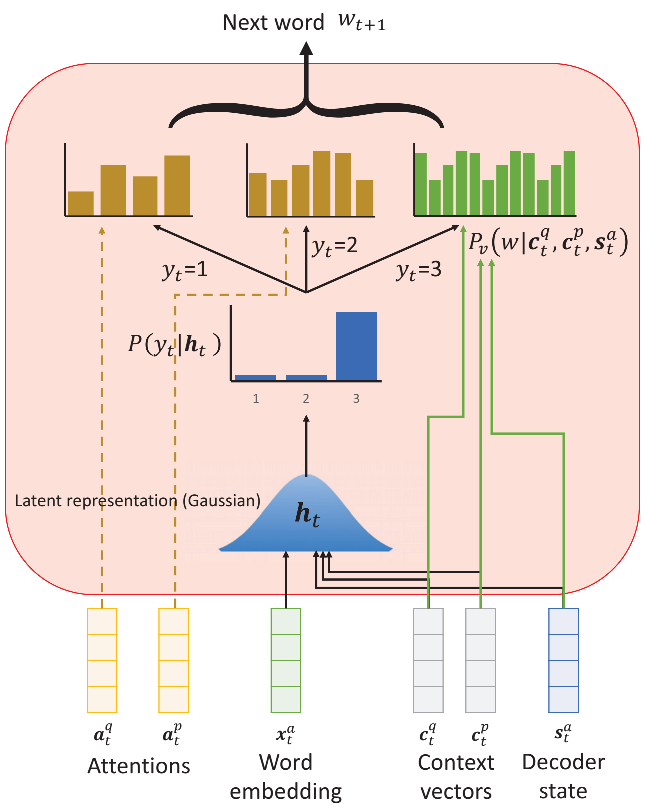

Figure 2: An overview of the stochastic selector network in timestep t. It takes the outputs of the preceding components in LatentQA, and produces the next word in the answer to be generated. Best viewed in color.

Stochastic Selector Networks

Figure 2 illustrates how the stochastic selector network works in one timestep during decoding. In each timestep t, a stochastic selector network is used as a three-way switch to select a word from one of the three distributions: 1. Attention distribution $\mathbf{a}_t^q \in \mathbb{R}^{N_q}$ over question words (Equation 1), where N_q denotes the number of distinct words in the question. 2. Attention distribution $\mathbf{a}_t^p \in \mathbb{R}^{N_p}$ over paragraph words (Equation 2), where N_p denotes the number of distinct words in the paragraph. 3. Conditional vocabulary distribution $P_v(w|\mathbf{c}_t^q, \mathbf{c}_t^p, \mathbf{s}_t^a)$ over all words in the vocabulary, which is obtained by:

$$P_v(w|\mathbf{c}_t^q, \mathbf{c}_t^p, \mathbf{s}_t^a) = \text{softmax}(\mathbf{W}^v \cdot [\mathbf{c}_t^q, \mathbf{c}_t^p, \mathbf{s}_t^a] + \mathbf{b}^v), \quad (4)$$

where $\mathbf{c}_t^q$ and $\mathbf{c}_t^p$ are context vectors, and $\mathbf{s}_t^a$ is a decoder state. $\mathbf{W}^v$ and $\mathbf{b}^v$ are learnable parameters.

To determine which of the three distributions a new word w_{t+1} is selected from, we introduce a discrete latent variable $y_t \in \{1, 2, 3\}$ as an indicator. The word w_{t+1} is then generated from the distribution $P(w_{t+1}|y_t)$ given by:

$$P(w_{t+1}|y_t) = \begin{cases} \sum_{i:w_i=w_{t+1}} a_{ti}^q & y_t = 1 \\ \sum_{i:w_i=w_{t+1}} a_{ti}^p & y_t = 2 \\ P_v(w|\mathbf{c}_t^q, \mathbf{c}_t^p, \mathbf{s}_t^a) & y_t = 3. \end{cases} \quad (5)$$

The random variable y_t follows a distribution $P(y_t|\mathbf{h}_t)$ conditioned on the latent representation $\mathbf{h}_t \in \mathbb{R}^{N_h}$ that models the interactions among question q, paragraph p and decoding word w_t as a stochastic vector.

In the LatentQA model, we choose the form of $\mathbf{h}_t$ to be a parameterized isotropic Gaussian:

$$\mathbf{h}_t|\mathbf{v}_t \sim \mathcal{N}(\mathbf{h}_t|\boldsymbol{\mu}_\theta(\mathbf{v}_t), \boldsymbol{\sigma}^2_\theta(\mathbf{v}_t)), \tag{6}$$

$$\mathbf{v}_t = [\mathbf{c}^q_t, \mathbf{c}^p_t, \mathbf{s}^a_t, \mathbf{x}^a_t], \tag{7}$$

$$\boldsymbol{\mu}_\theta(\mathbf{v}_t) = \mathrm{SLP}_{\theta 1}(\mathbf{v}_t), \log \boldsymbol{\sigma}_\theta(\mathbf{v}_t) = \mathrm{SLP}_{\theta 2}(\mathbf{v}_t), \tag{8}$$

where $\mathbf{x}^a_t$ is the embedding of the answer word in timestep t. $\mathrm{SLP}_{\theta 1}(\cdot)$ and $\mathrm{SLP}_{\theta 2}(\cdot)$ are two single-layer perceptrons with the tanh activation.

Compared with its deterministic counterpart $\mathbf{v}_t$, the stochastic representation $\mathbf{h}_t$ models the uncertainty inherent in questions, paragraphs and answers. For example, a single question can have multiple interpretations, and thus two individuals can provide very different answers to this question. In addition to subjective interpretations, uncertainty comes from the fact of answers being abstractions that summarize relevant and prominent information by leaving out less important message.

Moreover, by marginalizing over discrete variable y_t and continuous vector $\mathbf{h}_t$, the stochastic selector network has a natural safeguard against overfitting. This robustness enables the LatentQA model to perform well on a small QA training dataset.

Unlike prior QFS models which need source labels for training (Hasselqvist, Helmertz, and Kågebäck 2017), LatentQA models the sources of words as latent variables, and thus learns from data to infer their values. In this way, the sources can be determined dynamically based on generation states. The inferred values reveal the source of every word in a generated answer, and thus allow us to visualize where every answer word comes from.

Learning Model Parameters

Objective

To learn the parameters θ in LatentQA with latent variables, we maximize the marginal log-likelihood of words in all answers:

$$\log P_\theta(\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \cdots, \mathbf{w}^{(N)}) = \sum_{i=1}^{N} \log P_\theta(\mathbf{w}^{(i)}). \tag{9}$$

Unfortunately, direct optimization of this marginal is intractable, so we approximate it by variational inference (Jordan et al. 1999), and use the variational lower bound as the maximization objective. For the ith instance, the lower bound is given as: (superscript $\cdot^{(i)}$ is omitted for simplicity.)

$$\begin{aligned}&\log P_\theta(\mathbf{w})\\ &\geq \mathbb{E}_{Q_\phi}[\log P_\theta(\mathbf{w}|\mathbf{h})] - \mathrm{KL}(Q_\phi(\mathbf{h}|\mathbf{v}, \mathbf{w}')||P_\theta(\mathbf{h}|\mathbf{v}))\\ &:= \mathcal{L},\end{aligned} \tag{10}$$

where $Q_\phi(\mathbf{h}|\mathbf{v}, \mathbf{w}')$ is an approximate posterior distribution parameterized by ϕ, which avoids having to solve the intractable true posterior. $\mathbf{w}'$ denotes the sequence of answer words that the decoder targets in each timestep, meaning that each word in $\mathbf{w}'$ is one step ahead of the corresponding one in $\mathbf{w}$: $w'_t = w_{t+1}$. The KL-divergence term encourages the approximate posterior $Q_\phi(\mathbf{h}|\mathbf{v}, \mathbf{w}')$ to be close to the prior $P_\theta(\mathbf{h}|\mathbf{v})$ defined in Equation 6.

The posterior $Q_\phi(\mathbf{h}_t|\mathbf{v}_t, w_{t+1})$ in timestep t, analogously to $P_\theta(\mathbf{h}_t|\mathbf{v}_t)$, is defined as another isotropic Gaussian distribution parameterized by two different single-layer perceptrons:

$$\begin{aligned}&Q_\phi(\mathbf{h}_t|\mathbf{v}_t, w_{t+1}) =\\ &\qquad \mathcal{N}(\mathbf{h}_t|\boldsymbol{\mu}_\phi(\mathbf{v}_t, w_{t+1}), \boldsymbol{\sigma}^2_\phi(\mathbf{v}_t, w_{t+1})),\end{aligned} \tag{11}$$

$$\mathbf{r}_t = \mathrm{linear}_\phi(\mathrm{one\_hot}(w_{t+1})), \tag{12}$$

$$\boldsymbol{\mu}_\phi(\mathbf{v}_t, w_{t+1}) = \mathrm{SLP}_{\phi 3}([\mathbf{v}_t, \mathbf{r}_t]), \tag{13}$$

$$\log \boldsymbol{\sigma}_\phi(\mathbf{v}_t, w_{t+1}) = \mathrm{SLP}_{\phi 4}([\mathbf{v}_t, \mathbf{r}_t]), \tag{14}$$

where $\mathbf{r}_t$ is a linear transformation of the one-hot representation of target word w_{t+1}. Parameterizing $Q_\phi(\mathbf{h}_t|\mathbf{v}_t, w_{t+1})$ by $\mathbf{r}_t$ gives the posterior distribution conditioned on the target label, which enables Bayesian inference of the LatentQA model.

In the lower bound (10), we analytically integrate the KL-divergence between two non-standard isotropic Gaussians, which gives:

$$\begin{aligned}&\mathrm{KL}(Q_\phi(\mathbf{h}|\mathbf{v}, \mathbf{w}')||P_\theta(\mathbf{h}|\mathbf{v}))\\ &= \sum_t^{N_a} \sum_j^{N_h} \left\{ \log \frac{\sigma_{\theta tj}}{\sigma_{\phi tj}} + \frac{1}{2}\left[\frac{(\mu_{\phi tj} - \mu_{\theta tj})^2}{\sigma^2_{\theta tj}} + \frac{\sigma^2_{\phi tj}}{\sigma^2_{\theta tj}} - 1 \right] \right\},\end{aligned} \tag{15}$$

where N_a is the number of answer words, and N_h is the dimensionality of latent representation $\mathbf{h}$. Our derivation of the KL-divergence is detailed in the Appendix.

To estimate the expectation term $\mathbb{E}_{Q_\phi}$ in the lower bound (10), we use the reparameterization trick for variational Bayesian methods (Kingma and Welling 2014), and reparameterize $\mathbf{h}_t = \boldsymbol{\mu}_t + \boldsymbol{\sigma}_t \cdot \boldsymbol{\epsilon}_t$, where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \mathbf{I})$, for the gradients w.r.t. both θ and ϕ. This trick reduces the variance in stochastic estimation.

Inferring Discrete Latent Variables

The LatentQA model contains discrete latent variables $\mathbf{y}$, which presents a challenge to backpropagation through samples from the conditional distribution $P(y_t|\mathbf{h}_t)$. To address this problem, we create a differentiable estimator for discrete random variables with the Gumbel-Softmax trick (Jang, Gu, and Poole 2017).

In particular, we first compute the discrete distribution $P(y_t|\mathbf{h}_t)$ with three class probabilities π_1, π_2, π_3 by:

$$P(y_t|\mathbf{h}_t) = \mathrm{softmax}(\mathrm{linear}(\mathbf{h}_t)). \tag{16}$$

The Gumbel-Max trick (Gumbel 1954) allows us to draw samples from the discrete distribution $P(y_t|\mathbf{h}_t)$ by calculating $\mathrm{one\_hot}(\arg\max_i[g_i + \log \pi_i])$, where g_1, g_2, g_3 are i.i.d. samples drawn from the $\mathrm{Gumbel}(0, 1)$ distribution. For the inference of a discrete variable y_t, we approximate the Gumbel-Max trick by the continuous softmax function (in place of $\arg\max$) with temperature τ to generate a sample vector $\hat{\mathbf{y}}_t$:

$$\hat{y}_{ti} = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{3} \exp((\log(\pi_j) + g_j)/\tau)}. \tag{17}$$

When τ approaches zero, the generated sample $\hat{\mathbf{y}}_t$ becomes a one-hot vector. τ is gradually annealed over the course of training.

This new differentiable estimator allows us to backpropagate through $y_t \sim P(y_t|\mathbf{h}_t)$ for gradient estimation of every single sample. In our experiments, we train LatentQA using Adagrad (Duchi, Hazan, and Singer 2011) with a learning rate of 0.15 and an initial accumulator value of 0.1.

Experiments

In our experiments, we compare LatentQA with the state-of-the-art models that generate abstractive answers, as well as the ablations of LatentQA. In addition, we illustrate how well-formed answers are generated by LatentQA. Finally, we analyze a couple of sample answers generated by LatentQA.

Datasets and Evaluation Metrics

We conduct our experiments on two public benchmark datasets of natural answer generation, MARCO (Nguyen et al. 2016) and DuReader (He et al. 2018).

In the latest MARCO V2.1 dataset, the questions are user queries issued to the Bing search engine and the contextual paragraphs are from real web documents. The data has been split into a training set (154K QA pairs), a dev set (12K QA pairs) and a test set (101K questions with unpublished answers). DuReader is the largest Chinese document reading comprehension dataset, which contains 272K QA pairs in the training set, 10K QA pairs in the dev set and 20K questions in the test set. In both benchmark datasets, the well-formed answers are used for training. Since true well-formed answers are not available in the test sets of both benchmarks, we hold out the dev sets for evaluation in our experiments, and test models for each question on its associated paragraphs by concatenating them all together. We tune the hyper-parameters by cross-validation on the training sets.

The answers in the datasets are human-generated and not necessarily sub-spans of the paragraphs. We use the official evaluation tools of MARCO and DuReader, which compute metrics BLEU-1 (Papineni et al. 2002) and ROUGE-L (Lin 2004) for MARCO, and compute BLEU-4 and ROUGE-L for DuReader.

Implementation Details

In LatentQA, we use 300-dimensional pre-trained *Glove* word embeddings (Pennington, Socher, and Manning 2014) for initialization with update during training. The dimension of hidden states is set to 256 for every LSTM. The latent representation $\mathbf{h}_t$ has $N_h = 100$ dimensions. We use a vocabulary of 50K words (filtered by frequency). Note that stochastic selector networks enables LatentQA to handle out-of-vocabulary words by allowing an answer word to come from the paragraph or the question.

Model	Rouge-L	Bleu-1
BiDAF	19.42	13.03
BiDAF+Seq2Seq	34.15	29.68
S-Net	42.71	36.19
S-Net+Seq2Seq	46.83	39.74
gQA	45.75	41.10
QFS	40.58	39.96
VNET	45.93	41.02
LatentQA	**50.97**	**45.48**

Table 2: Metrics of LatentQA and state-of-the-art QA models on the MARCO dataset.

At both training and test stages, we truncate a paragraph to 800 words, and limit the length of an answer to 120 words. We train on a single Tesla M40 GPU with the batch size of 16. At test time, answers are generated using beam search with the beam size of 4.

Comparison with State-of-the-Art QA Models

We compare LatentQA with the following state-of-the-art QA models:

1. **BiDAF** (Seo et al. 2017): A multi-stage hierarchical process that represents context at different levels of granularity, and using the bi-directional attention flow mechanism for answer extraction.
2. **BiDAF+Seq2Seq**: A BiDAF model followed by an additional sequence-to-sequence model for answer generation.
3. **S-Net** (Tan et al. 2018): An extraction-then-synthesis framework to synthesize answers from extracted evidences.
4. **S-Net+Seq2Seq**: An S-Net model followed by an additional sequence-to-sequence model for answer generation.
5. **gQA** (Mitra 2017): A generative approach to question answering by incorporating the copying mechanism (from paragraphs only) and the coverage vector.
6. **QFS** (Nema et al. 2017): A model that adapts the query-focused summarization model to answer generation.
7. **VNET** (Wang et al. 2018): An MRC model that enables answer candidates from different paragraphs to verify each other based on their content representations.

Table 2 shows the comparison of QA models in Rouge-L and Bleu-1. Abstractive QA models (e.g., LatentQA) are superior to extractive models (e.g., BiDAF) in answer quality according to the table. As an example, BiDAF answers a question with a short span of text extracted from the paragraph. Such an answer extraction model is unable to produce a long enough summary as an answer. BiDAF+Seq2Seq produces better answers than extractive BiDAF does by incorporating an additional sequence-to-sequence model for answer generation. LatentQA outperforms all the other QA models by a large margin without the need to build the extraction model as BiDAF+Seq2Seq and S-Net+Seq2Seq do. Instead, the LatentQA model generates natural answers with

Model	Syntactic	Correct	Well-formed
BiDAF	**4.31**	3.68	2.66
BiDAF+Seq2Seq	3.84	3.15	3.22
S-Net	3.90	3.87	2.73
S-Net+Seq2Seq	3.98	3.22	3.50
gQA	3.78	3.54	3.13
QFS	3.65	3.39	2.87
VNET	4.16	3.72	3.11
LatentQA	4.22	**4.09**	**4.61**

Table 3: Human evaluation of LatentQA and state-of-the-art QA models on the MARCO dataset. Scores range in $[1, 5]$.

some words selected from questions, beyond the capability of the other models.

Since neither Rouge-L nor Bleu-1 can measure the quality of generated answers in terms of their correctness and accuracy, we also conduct human evaluation on Amazon Mechanical Turk. The evaluation assesses the answer quality on grammaticality, correctness and well-formedness. We randomly select 100 questions from the MARCO dev set, and ask turkers for ratings in a Likert scale ($\in [1, 5]$) on the generated answers.

Table 3 reports the human evaluation scores of LatentQA and compared models. The LatentQA model outperforms all the others in generating correct and well-formed answers. In particular, extractive models, such as BiDAF, are inferior in well-formedness, since the answers are often short pieces of text extracted from paragraphs. On the other hand, BiDAF performs the best among all the models in syntactic correctness. This is not surprising, as it is easier to make short answers grammatically correct. Nevertheless, LatentQA still beats the rest of baselines in terms of having better grammar.

Table 4 reflects the comparison results on the DuReader dataset in Rouge-L and Bleu-4. LatentQA performs better than all the others do, as shown in this table. The superiority of LatentQA on DuReader confirms not only the effectiveness of LatentQA's extension to the sequence-to-sequence model and its stochastic selector networks, but also LatentQA's generalizability to a language other than English (i.e., Chinese) in generating high-quality well-formed answers.

Model	Rouge-L	Bleu-4
BiDAF	27.22	21.53
BiDAF+Seq2Seq	32.89	28.67
S-Net	41.60	38.32
S-Net+Seq2Seq	45.84	43.35
gQA	45.73	43.91
QFS	38.87	36.43
VNET	46.09	43.56
LatentQA	**49.16**	**47.20**

Table 4: Metrics of LatentQA and state-of-the-art QA models on the DuReader dataset.

Ablation	Rouge-L	Bleu-1
Full LatentQA	**50.97**	**45.48**
✗ question source	48.76	43.02
✗ stochastic representation	48.51	42.87
✗ 3-way discrete switch	48.29	43.04
✗ full selector network	47.36	40.61

Table 5: Ablation tests of LatentQA on the MARCO dataset.

Ablation Studies

We conduct ablation studies to assess the individual contribution of every component in LatentQA. Table 5 reports the performance of the full LatentQA model and its ablations on the MARCO dataset.

We evaluate how much selecting words from a question contributes to well-formed answer constitution by removing the question source from the three-way selector, and retaining the paragraph and vocabulary sources. The question source turns out to play an important role in generating well-formed answers, with a drop to 48.76 on Rouge-L after the question source is removed. For ablating the stochastic representation, we replace it with the deterministic representation. The stochastic representation proves to be critical with a drop of about 5% on both metrics after the replacement.

The three-way discrete switch accounts for over 5% of performance degradation from full LatentQA, which clearly demonstrates the superiority of the discrete module over the continuous counterpart and the power of discrete Bayesian inference and the three-way selection. Finally, we ablate full stochastic selector networks, which effectively leads to a sequence-to-sequence model with the copying mechanism from paragraphs and questions (pointer-generator). This ablation results in a significant drop in Rouge-L to 47.36, confirming the superiority of stochastic selector networks over vanilla pointer-generator in leveraging the question source to generate well-formed answers.

Visualization

The stochastic selector networks allow us to visualize how every word in an answer is generated from one of the sources of the question, paragraph and vocabulary, which gives us insights about how LatentQA works.

Table 6 visualizes the sample answers generated by LatentQA and the source every answer word is selected from. The first question leads to a Yes/No answer. This type of question goes beyond the questions that an extractive model can handle, since the paragraph may not contain the word *Yes/No* to be extracted. In contrast, by reading through and summarizing the paragraph, LatentQA gives a Yes answer correctly, and generates a well-formed answer with supplementary context. In generating the answer, it first picks word *Yes* from the vocabulary for its high source probability (dark cyan). The model then completes the well-formed answer with a supplementary sentence (e.g., words *kill someone underwater* obtained from the question), which clearly demonstrates the significant contribution of words selected from questions in making natural answers.

Question 1	
Can bullets kill someone underwater?	
Answer with source probabilities	
Question src	Yes, bullets can kill someone underwater.
Paragraph src	Yes, bullets can kill someone underwater.
Vocabulary src	Yes, bullets can kill someone underwater.
Answer colored by source	
Yes, bullets can kill someone underwater.	

Question 2	
What are causes of insomnia in women?	
Answer with source probabilities	
Question	Fatigue and stress are the causes of insomnia in women.
Paragraph	Fatigue and stress are the causes of insomnia in women.
Vocabulary	Fatigue and stress are the causes of insomnia in women.
Answer colored by source	
Fatigue and stress are the causes of insomnia in women.	

Table 6: Visualizations of sample answers and the source of individual words in the answers. The **Answer with source probabilities** section displays a heatmap on answer words selected from the question, paragraph and vocabulary, respectively. A slot with a higher source probability is highlighted in darker cyan. The **Answer colored by source** section shows the answer in which every word is colored based on the source it was actually selected from. Words in blue come from the question, words in red come from the paragraph, and words in green come from the vocabulary. The visualizations are best viewed in color.

Different from the first question, the second one is an open-ended question. To answer this question, LatentQA selects some words from every source based on their selection probabilities. From the table, it can be seen that in the answer the keywords *fatigue* and *stress* come from the paragraph source. This results from reading comprehension of the model on the paragraph. By contrast, the question source has the other content words *causes*, *insomnia* and *women* with high source probabilities, which leads the model to form the answer with the content words from the question. To make a complete sentence, the model selects the filler words *and*, *are*, *the*, *of* and *in* from the vocabulary. This leads to a final answer in good form, which is both semantically correct and comprehensive.

Conclusion and Future Work

This paper introduces a new neural model LatentQA that is designed for well-formed answer generation. With stochastic selector networks, the model determines which of the question, paragraph and vocabulary to select every word from based on its selection probability in generating an answer. The LatentQA model can also be extended to integrate external knowledge by generating answer words from extra sources, such as knowledge bases, which we will explore further in future research.

Appendix

The KL-divergence between the two non-standard isotropic Gaussians $Q_\phi(\mathbf{h}|\mathbf{v},\mathbf{w}')$ and $P_\theta(\mathbf{h}|\mathbf{v})$ is given by:

$$\begin{aligned}
&\mathrm{KL}(Q_\phi(\mathbf{h}|\mathbf{v},\mathbf{w}')||P_\theta(\mathbf{h}|\mathbf{v}))\\
&=\sum_t \int (\log Q_{\phi t} - \log P_{\theta t}) Q_{\phi t}\, d\mathbf{h_t}\\
&=\sum_t \int \frac{1}{2}\left[\log\frac{|\mathbf{\Sigma}_{\theta t}|}{|\mathbf{\Sigma}_{\phi t}|} - (\mathbf{h}_t-\boldsymbol{\mu}_{\phi t})^\intercal \mathbf{\Sigma}_{\phi t}^{-1}(\mathbf{h}_t-\boldsymbol{\mu}_{\phi t})\right.\\
&\quad\left.+(\mathbf{h}_t-\boldsymbol{\mu}_{\theta t})^\intercal \mathbf{\Sigma}_{\theta t}^{-1}(\mathbf{h}_t-\boldsymbol{\mu}_{\theta t})\right] Q_{\phi t}\, d\mathbf{h_t}\\
&=\sum_t \frac{1}{2}\left\{\log\frac{|\mathbf{\Sigma}_{\theta t}|}{|\mathbf{\Sigma}_{\phi t}|} - \mathrm{tr}\left\{\mathrm{E}\left[(\mathbf{h}_t-\boldsymbol{\mu}_{\phi t})(\mathbf{h}_t-\boldsymbol{\mu}_{\phi t})^\intercal\right]\right.\right.\\
&\quad\left.\left.\mathbf{\Sigma}_{\phi t}^{-1}\right\} + \mathrm{E}\left[(\mathbf{h}_t-\boldsymbol{\mu}_{\theta t})^\intercal \mathbf{\Sigma}_{\theta t}^{-1}(\mathbf{h}_t-\boldsymbol{\mu}_{\theta t})\right]\right\}\\
&=\sum_t \frac{1}{2}\left\{\log\frac{|\mathbf{\Sigma}_{\theta t}|}{|\mathbf{\Sigma}_{\phi t}|} - \mathrm{tr}\left\{\mathbf{I}\right\} + (\boldsymbol{\mu}_\phi-\boldsymbol{\mu}_\theta)^\intercal \mathbf{\Sigma}_\theta^{-1}\right.\\
&\quad\left.(\boldsymbol{\mu}_\phi-\boldsymbol{\mu}_\theta) + \mathrm{tr}\left\{\mathbf{\Sigma}_\theta^{-1}\mathbf{\Sigma}_\phi\right\}\right\}\\
&=\sum_t\sum_j\left\{\log\frac{\sigma_{\theta tj}}{\sigma_{\phi tj}} + \frac{1}{2}\left[\frac{(\mu_{\phi tj}-\mu_{\theta tj})^2}{\sigma_{\theta tj}^2} + \frac{\sigma_{\phi tj}^2}{\sigma_{\theta tj}^2} - 1\right]\right\},
\end{aligned}$$

where $\mathbf{\Sigma}_t = \mathrm{diag}(\boldsymbol{\sigma}_t^2)$ is a square diagonal matrix with elements of vector $\boldsymbol{\sigma}_t^2$ on the diagonal.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* 12:2121–2159.

Fu, Y., and Feng, Y. 2018. Natural answer generation with heterogeneous memory. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 185–195.

Gao, S.; Ren, Z.; Zhao, Y.; Zhao, D.; Yin, D.; and Yan, R. 2019. Product-aware answer generation in e-commerce question-answering. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*.

Gumbel, E. 1954. *Statistical theory of extreme values and some practical applications: a series of lectures*. Applied mathematics series. U. S. Govt. Print. Office.

Hasselqvist, J.; Helmertz, N.; and Kågebäck, M. 2017. Query-based abstractive summarization using neural networks. *CoRR* abs/1712.06100.

He, S.; Liu, C.; Liu, K.; and Zhao, J. 2017. Generating natural answers by incorporating copying and retrieving mechanisms in sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 199–208.

He, W.; Liu, K.; Liu, J.; Lyu, Y.; Zhao, S.; Xiao, X.; Liu, Y.; Wang, Y.; Wu, H.; She, Q.; Liu, X.; Wu, T.; and Wang, H. 2018. DuReader: a Chinese machine reading comprehension dataset from real-world applications. In *Proceedings of the Workshop on Machine Reading for Question Answering*, 37–46. Melbourne, Australia: Association for Computational Linguistics.

Jang, E.; Gu, S.; and Poole, B. 2017. Categorical reparametrization with gumbel-softmax. In *Proceedings of the International Conference on Learning Representations*.

Jordan, M. I.; Ghahramani, Z.; Jaakkola, T. S.; and Saul, L. K. 1999. An introduction to variational methods for graphical models. *Mach. Learn.* 37(2):183–233.

Kingma, D., and Welling, M. 2014. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations*.

Lin, C.-Y. 2004. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the ACL Workshop: Text Summarization Braches Out 2004*, 10.

Mitra, R. 2017. An abstractive approach to question answering. *CoRR* abs/1711.06238.

Nallapati, R.; Zhou, B.; dos Santos, C.; Gulcehre, C.; and Xiang, B. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*.

Nema, P.; Khapra, M. M.; Laha, A.; and Ravindran, B. 2017. Diversity driven attention model for query-based abstractive summarization. In *Proceedings of the 55th Annual Meeting of the ACL*, 1063–1072.

Nguyen, T.; Rosenberg, M.; Song, X.; Gao, J.; Tiwary, S.; Majumder, R.; and Deng, L. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS)*.

Nishida, K.; Saito, I.; Nishida, K.; Shinoda, K.; Otsuka, A.; Asano, H.; and Tomita, J. 2019. Multi-style generative reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2273–2284. Florence, Italy: Association for Computational Linguistics.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, 311–318.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, 1532–1543.

Peters, M.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2227–2237. New Orleans, Louisiana: Association for Computational Linguistics.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2383–2392.

Rajpurkar, P.; Jia, R.; and Liang, P. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 784–789. Association for Computational Linguistics.

See, A.; Liu, P. J.; and Manning, C. D. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Seo, M.; Kembhavi, A.; Farhadi, A.; and Hajishirzi, H. 2017. Bidirectional attention flow for machine comprehension. In *Proceedings of the International Conference on Learning Representations*.

Tan, C.; Wei, F.; Yang, N.; Du, B.; Lv, W.; and Zhou, M. 2018. S-net: From answer extraction to answer synthesis for machine reading comprehension. In *AAAI*.

Tao, C.; Gao, S.; Shang, M.; Wu, W.; Zhao, D.; and Yan, R. 2018. Get the point of my utterance! learning towards effective responses with multi-head attention mechanism. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 4418–4424.

Wang, Y.; Liu, K.; Liu, J.; He, W.; Lyu, Y.; Wu, H.; Li, S.; and Wang, H. 2018. Multi-passage machine reading comprehension with cross-passage answer verification. In *Proceedings of the 56th Annual Meeting of the ACL*, 1918–1927.

Wang, W.; Yan, M.; and Wu, C. 2018. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. In *Proceedings of the 56th Annual Meeting of the ACL*, 1705–1714.

Yan, M.; Xia, J.; Wu, C.; Bi, B.; Zhao, Z.; Zhang, J.; Si, L.; Wang, R.; Wang, W.; and Chen, H. 2019. A deep cascade model for multi-document reading comprehension. In *AAAI Conference on Artificial Intelligence*.

Yin, J.; Jiang, X.; Lu, Z.; Shang, L.; Li, H.; and Li, X. 2016. Neural generative question answering. In *Proceedings of the Workshop on Human-Computer Question Answering*, 36–42.