Mixture Content Selection for Diverse Sequence Generation

Jaemin Cho[1]* **Minjoon Seo**[2,3] **Hannaneh Hajishirzi**[1,3]
Allen Institute for AI[1] Clova AI, NAVER[2] University of Washington[3]
heythisischo@gmail.com {minjoon,hannaneh}@cs.washington.edu

Abstract

Generating diverse sequences is important in many NLP applications such as question generation or summarization that exhibit semantically one-to-many relationships between source and the target sequences. We present a method to explicitly separate diversification from generation using a general plug-and-play module (called SELECTOR) that wraps around and guides an existing encoder-decoder model. The diversification stage uses a mixture of experts to sample different binary masks on the source sequence for diverse content selection. The generation stage uses a standard encoder-decoder model given each selected content from the source sequence. Due to the non-differentiable nature of discrete sampling and the lack of ground truth labels for binary mask, we leverage a proxy for ground-truth mask and adopt stochastic hard-EM for training. In question generation (SQuAD) and abstractive summarization (CNN-DM), our method demonstrates significant improvements in accuracy, diversity and training efficiency, including state-of-the-art top-1 accuracy in both datasets, 6% gain in top-5 accuracy, and 3.7 times faster training over a state-of-the-art model. Our code is publicly available at https://github.com/clovaai/FocusSeq2Seq.

1 Introduction

Generating target sequences given a source sequence has applications in a wide range of problems in NLP with different types of relationships between the source and target sequences. For instance, paraphrasing or machine translation exhibit a one-to-one relationship because the source and the target should carry the same meaning. On the other hand, summarization or question generation exhibit one-to-many relationships because a single source often results in diverse target sequences with different semantics. Fig. 1 shows different questions that can be generated from a given passage.

Source Passage: in december 1878 , tesla left graz and severed all relations with his family to hide the fact that he dropped out of school .
Target: what did tesla do in december 1878?

Focus 1: in december 1878 , tesla left graz and severed all relations with his family to hide the fact that he dropped out of school.
(Ours) ⇒ what did tesla do?

Focus 2: in december 1878 , tesla left graz and severed all relations with his family to hide the fact that he dropped out of school.
(Ours) ⇒ what did tesla do in december 1878?

Focus 3: in december 1878 , tesla left graz and severed all relations with his family to hide the fact that he dropped out of school .
(Ours) ⇒ what did tesla do to hide he dropped out of school?

Figure 1: Sample questions produced by our method from given passage-answer pair (answer is underlined). Our method generates diverse questions, by selecting different tokens to focus (colored) in contrast to 3-mixture decoder (Shen et al., 2019) that generates 3 identical questions: "what did tesla do to hide the fact that he dropped out of school?".

Encoder-decoder models (Cho et al., 2014) are widely used for sequence generation, most notably in machine translation where neural models are now often almost as good as human translators in some language pairs. However, a standard encoder-decoder often shows a poor performance when it attempts to produce multiple, diverse outputs. Most recent methods for diverse sequence generation leverage diversifying decoding steps through alternative search algorithms (Fan et al., 2018; Vijayakumar et al., 2018) or mixture of decoders (He et al., 2018; Shen et al., 2019). These methods promote diversity at the decoding

*Most work done during internship at Clova AI.

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 3121–3131, Hong Kong, China, November 3–7, 2019.

step, while a more focused selection of the source sequence can lead to diversifying the semantics of the generated target sequences.

In this paper, we present a method for diverse generation that separates diversification and generation stages. The diversification stage leverages content selection to map the source to multiple sequences, where each mapping is modeled by *focusing* on different tokens in the source (one-to-many mapping). The generation stage uses a standard encoder-decoder model to generate a target sequence given each selected content from the source (one-to-one mapping). We present a generic module called SELECTOR that is specialized for diversification. This module can be used as a plug-and-play to an arbitrary encoder-decoder model for generation without architecture change.

The SELECTOR module leverages a mixture of experts (Jacobs et al., 1991; Eigen et al., 2014) to identify diverse key contents to focus on during generation. Each mixture samples a sequential latent variable modeled as a binary mask on every source sequence token. Then an encoder-decoder model generates multiple target sequences given these binary masks along with the original source tokens. Due to the non-differentiable nature of discrete sampling, we adopt stochastic hard-EM for training SELECTOR. To mitigate the lack of ground truth annotation for the mask (content selection), we use the overlap between the source and target sequences as a simple proxy for the ground-truth mask.

We experiment on question generation and abstractive summarization tasks and show that our method achieves the best trade-off between accuracy and diversity over previous models on SQuAD (Rajpurkar et al., 2016) and CNN-DM (Hermann et al., 2015; Nallapati et al., 2016; See et al., 2017) datasets. In particular, compared to the recently-introduced mixture decoder (Shen et al., 2019) that also aims to diversify outputs by creating multiple decoders, our modular method not only demonstrates better accuracy and diversity, but also trains 3.7 times faster.

2 Related Work

Diverse Search Algorithms Beam search, the most commonly used search algorithm for decoding, is known to produce samples that are short, contain repetitive phrases, and share majority of their tokens. Hence several methods are introduced to diversify search algorithms for decoding. Graves (2013); Chorowski and Jaitly (2017) tune temperature hyperparameter in softmax function. Vijayakumar et al. (2018); Li et al. (2016b) penalize similar samples during beam search in order to obtain diverse set of samples. Cho (2016) adds random noise to RNN decoder hidden states. Fan et al. (2018) sample tokens from top-k tokens at each decoding step. Our method is orthogonal to these search-based strategies, in that they diversify *decoding* while our method diversifies which content to be focused during *encoding*. Moreover, our empirical results show diversification with stochastic sampling hurts accuracy significantly.

Deep Mixture of Experts Several methods adopt a deep mixture of experts (MoE) (Jacobs et al., 1991; Eigen et al., 2014) to diversify decoding steps. Yang et al. (2018) introduce soft mixture of softmax on top of the output layer of RNN language model. He et al. (2018); Shen et al. (2019) introduce mixture of decoders with uniform mixing coefficient to improve diversity in machine translation. Among these, the closest to ours is the mixture decoder (Shen et al., 2019) that also adopts hard-EM for training, where a minimum-loss predictor is assigned to each data point, which is also known as multiple choice learning (Guzman-Rivera et al., 2012; Lee et al., 2016). While Shen et al. (2019) makes RNN decoder as a MoE, we make SELECTOR as a MoE to diversify content selection and let the encoder-decoder models one-to-one generation. As shown in our empirical results, our method achieves a better accuracy-diversity trade-off while reducing training time significantly.

Variational Autoencoders Variational Autoencoders (VAE) (Kingma and Welling, 2013) are used for diverse generation in several tasks, such as language modeling (Bowman et al., 2016), machine translation (Zhang et al., 2016; Su et al., 2018; Deng et al., 2018; Shankar and Sarawagi, 2019), and conversation modeling (Serban et al., 2017; Wen et al., 2017; Zhao et al., 2017; Park et al., 2018; Wen and Luong, 2018; Gu et al., 2019). These methods sample diverse latent variables from an approximate posterior distribution, but often suffer from a posterior collapse where the sampled latent variables are ignored (Bowman et al., 2016; Park et al., 2018; Kim et al., 2018b;

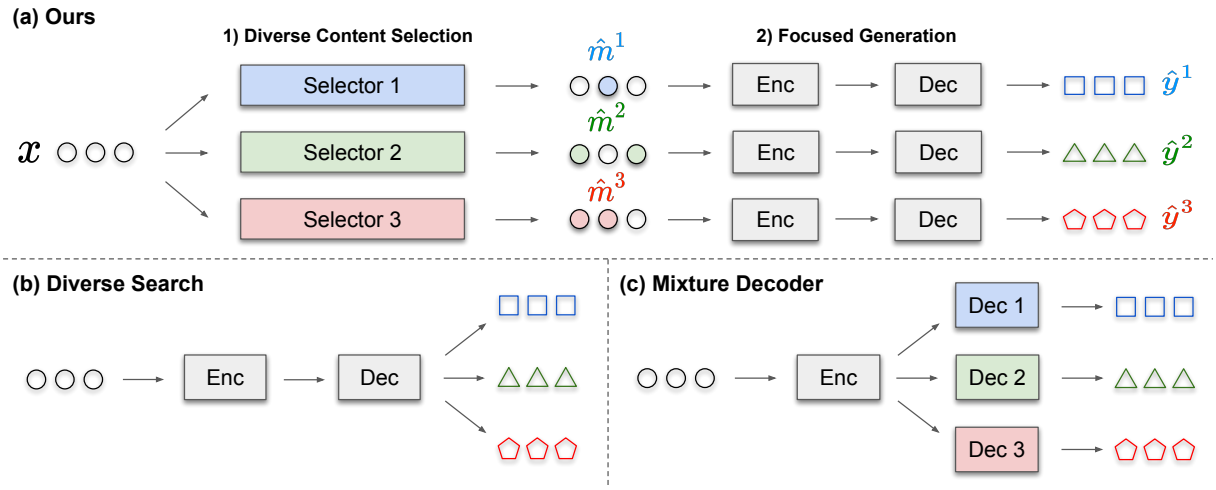

Figure 2: Overview of diverse sequence-to-sequence generation methods. (a) refers to our two-stage approach described throughout Section. 3, (b) refers to search-based methods (Vijayakumar et al., 2018; Li et al., 2016b; Fan et al., 2018), and (c) refers to mixture decoders (Shen et al., 2019; He et al., 2018).

Dieng et al., 2018; Xu and Durrett, 2018; He et al., 2019; Razavi et al., 2019). This is also observed in our initial experiments and Shen et al. (2019), where MoE-based methods significantly outperforms VAE-based methods because of the posterior collapse. Moreover, we observe that sampling mixtures makes training more stable compared to stochastic sampling of latent variables. Furthermore, our latent structure as a sequence of binary variables is different from most VAEs which use a fixed-size continuous latent variable. This gives a finer-grained control and interpretability on where to focus, especially when source sequence is long.

Diversity-Promoting Regularization Adding regularization to objective functions is used to diversify generation. Li et al. (2016a) introduce a term maximizing mutual information between source and target sentences. Chorowski and Jaitly (2017); Kalyan et al. (2018) introduce terms enforcing knowledge transfer among similar annotations. Our work is orthogonal to these methods and can potentially benefit from adding these regularization terms to our objective function.

Content Selection in NLP Selecting important parts of the context has been an important step in NLP applications (Reiter and Dale, 2000). Most recently, Ke et al. (2018); Min et al. (2018) conduct soft-/ hard-selection of key parts from source passages for question answering. Zhou et al. (2017b) use soft gating on the source document encoder for abstractive summarization. Li et al. (2018) guide abstractive summarization models with off-the-shelf keyword extractors. The most relevant work to ours are Subramanian et al. (2018) and Gehrmann et al. (2018). Subramanian et al. (2018) use a pointer network (Vinyals et al., 2015) for extracting key phrases for question generation and Gehrmann et al. (2018) use content selector to limit copying probability for abstractive summarization. The main purpose of these approaches is to enhance accuracy, while our method uses diverse content selection to enhance both accuracy and diversity (refer to our empirical results). Additionally, our method allows models to learn how to utilize information from the selected content, whereas Gehrmann et al. (2018) manually limit the copying mechanism on non-selected contents.

3 Method

In this paper, we focus on sequence generation tasks such as question generation and summarization that have one-to-many relationship. More formally, given a source sequence $\mathbf{x} = (x_1 \dots x_S) \in \mathcal{X}$, our goal is to model a conditional multimodal distribution for the target sequence $p(\mathbf{y}|\mathbf{x})$ that assigns high values on $p(\mathbf{y}^1|\mathbf{x}) \dots p(\mathbf{y}^K|\mathbf{x})$ for K valid mappings $\mathbf{x} \rightarrow \mathbf{y}^1 \dots \mathbf{x} \rightarrow \mathbf{y}^K$. For instance, Fig. 1 illustrates $K = 3$ different valid questions generated for a given passage.

For learning a multimodal distribution, it is not appropriate to use a generic encoder-decoder (Cho et al., 2014) that minimizes for the expected value of the log probability of all the valid mappings.

This can lead to a suboptimal mapping that is in the middle of the targets but not near any of them. As a solution, we propose to (1) introduce a latent variable called *focus* that factorizes the distribution into two stages, *select* and *generate* (Section 3.1), and (2) independently train the factorized distributions (Section 3.2).

3.1 Select and Generate

In order to factorize the multimodal distribution into the two stages (*select* and *generate*), we introduce a latent variable called *focus*. The intuition is that in the *select* stage we sample several meaningful *focus*, each of which indicates which part of the source sequence should be considered important. Then in the *generate* stage, each sampled focus biases the generation process towards being conditioned on the focused content.

Formally, we model *focus* with a sequence of binary variable, each of which corresponds to each token in the input sequence, i.e. $\mathbf{m} = \{m_1 \dots m_S\} \in \{0,1\}^S$. The intuition is that $m_t = 1$ indicates t-th source token x_t should be *focused* during sequence generation. For instance, in Fig. 1, colored tokens (green, red, or blue) show that different tokens are focused (i.e. values are 1) for different focus samples (out of 3). We first use the latent variable $\mathbf{m}$ to factorize $p(\mathbf{y}|\mathbf{x})$,

$$p(\mathbf{y}|\mathbf{x}) = \mathbb{E}_{\mathbf{m}\sim p_\phi(\mathbf{m}|\mathbf{x})}[p_\theta(\mathbf{y}|\mathbf{m},\mathbf{x})] \quad (1)$$

where $p_\phi(\mathbf{m}|\mathbf{x})$ is *selector* and $p_\theta(\mathbf{y}|\mathbf{x},\mathbf{m})$ is *generator*. The factorization separates focus selection from generation so that modeling multimodality (diverse outputs) can be solely handled in the *select* stage and *generate* stage can solely concentrate on the generation task itself. We now describe each component in more details.

Selector In order to directly control the diversity of the SELECTOR's outputs, we model it as a hard mixture of experts (hard-MoE) (Jacobs et al., 1991; Eigen et al., 2014), where each expert specializes in focusing on different parts of the source sequences. In Fig. 1 and 2, focus produced by each SELECTOR expert is colored differently. We introduce a multinomial latent variable $z \in \mathbb{Z}$, where $\mathbb{Z} = \{1 \dots K\}$, and let each focus $\mathbf{m}$ be assigned to one of K experts with uniform prior $p(z|\mathbf{x}) = \frac{1}{K}$. With this mixture setting, $p(\mathbf{m}|\mathbf{x})$ is recovered as follows.

$$\begin{aligned} p(\mathbf{m}|\mathbf{x}) &= \mathbb{E}_{z\sim p(z|\mathbf{x})}[p_\phi(\mathbf{m}|\mathbf{x},z)] \\ &= \frac{1}{K}\sum_{z}^{1\dots K} p_\phi(\mathbf{m}|\mathbf{x},z) \end{aligned} \quad (2)$$

We model SELECTOR with a single-layer Bi-directional Gated Recurrent Unit (Bi-GRU) (Cho et al., 2014) followed by two fully-connected layers and a Sigmoid activation. We feed (current hidden state $\mathbf{h}_t$, first and last hidden state $\mathbf{h}_1, \mathbf{h}_S$, and expert embedding $\mathbf{e}_z$) to a fully-connected layers (FC). Expert embedding $\mathbf{e}_z$ is unique for each expert and is trained from a random initialization. From our initial experiments, we found this parallel focus inference to be more effective than auto-regressive pointing mechanism (Vinyals et al., 2015; Subramanian et al., 2018). The distribution of the focus conditioned on the input $\mathbf{x}$ and the expert id z is the Bernoulli distribution of the resulting values,

$$\begin{aligned} (\mathbf{h}_1 \dots \mathbf{h}_S) &= \text{Bi-GRU}(\mathbf{x}) \\ o_t^z &= \sigma(\text{FC}([\mathbf{h}_t;\mathbf{h}_1;\mathbf{h}_S;\mathbf{e}_z])) \\ p_\phi(m_t|\mathbf{x},z) &= \text{Bernoulli}(o_t^z) \end{aligned} \quad (3)$$

To prevent low quality experts from *dying* during training, we let experts share all parameters, except for the individual expert embedding $\mathbf{e}_z$. We also reuse the word embedding of the generator in the word embedding of SELECTOR to promote cooperative knowledge sharing between SELECTOR and generator. With these parameter sharing techniques, adding a mixture of SELECTOR experts increases only a slight amount of parameters (GRU, FC, and $\mathbf{e}_z$) to sequence-to-sequence models.

Generator For maximum diversity, we sample one focus from each SELECTOR expert to approximate $p_\phi(\mathbf{m}|\mathbf{x})$. For a deterministic behavior, we threshold o_t^z with a hyperparameter th instead of sampling from the Bernoulli distribution. This gives us a list of focus $\mathbf{m}^1 \dots \mathbf{m}^K$ coming from K experts. Each focus $\mathbf{m}^z = (m_1^z \dots m_S^z)$ is encoded as embeddings and concatenated with the input embeddings of the source sequence $\mathbf{x} = (x_1 \dots x_S)$. An off-the-shelf generation function such as encoder-decoder can be used for modeling $p(\mathbf{y}|\mathbf{m},\mathbf{x})$, as long as it accepts a stream of input embeddings. We use an identical generation function with K different focus samples to produce K different diverse outputs.

Algorithm 1: Training
(N: Dataset size, K: Number of mixtures)

Data: $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{m}^{\text{guide (i)}})\}_{i=1}^{N}$

1 **for** $i \in \{1 \dots N\}$ **do**

/* Selector $p_\phi(\mathbf{m}|\mathbf{x}, z)$ E-step */

2 **for** $z \in \{1 \dots K\}$ **do**

3 $L_{\text{select}}^{(i)\ z} = -\log p_\phi(\mathbf{m}^{\text{guide }(i)}|\mathbf{x}^{(i)}, z)$

4 **end**

5 $z^{\text{best }(i)} = \underset{z}{\text{argmin}}\, L_{\text{select}}^{(i)\ z}$

/* Selector $p_\phi(\mathbf{m}|\mathbf{x}, z)$ M-step */

6 $\phi_{z^{\text{best }(i)}} = \phi_{z^{\text{best }(i)}} - \alpha \nabla_{\phi_{z^{\text{best }(i)}}} L_{\text{select}}^{(i)\ z^{\text{best }(i)}}$

/* Generator $p_\theta(\mathbf{y}|\mathbf{x}, \mathbf{m})$ Update */

7 $L^{(i)}{}_{\text{gen}} = -\log p_\theta(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}, \mathbf{m}^{\text{guide (i)}})$

8 $\theta = \theta - \alpha \nabla_\theta L_{\text{gen}}^{(i)}$

9 **end**

3.2 Training

Marginalizing the Bernoulli distribution in Eq. 3 by enumerating all possible focus is intractable since the cardinality of focus space 2^S grows exponentially with source sequence length S. Policy gradient (Williams, 1992; Yu et al., 2017) or Gumbel-softmax (Jang et al., 2017; Maddison et al., 2017) are often used to propagate gradients through a stochastic process, but we empirically found that these do not work well. We instead create *focus guide* and use it to independently and directly train the SELECTOR and the generator. Formally, a focus guide $\mathbf{m}^{\text{guide}} = (m_1^{\text{guide}} \dots m_S^{\text{guide}})$ is a simple proxy of whether a source token is *focused* during generation. We set t-th focus guide m_t^{guide} to 1 if t-th source token x_t is *focused* in target sequence $\mathbf{y}$ and 0 otherwise. During training, $\mathbf{m}^{\text{guide}}$ acts as a target for SELECTOR and is a given input for generator (teacher forcing). During inference, $\hat{\mathbf{m}}$ is sampled from SELECTOR and fed to the generator.

In question generation, we set m_t^{guide} to 1 if there is a target question token which shares the same word stem with passage token x_t. Then we set m_t^{guide} to 0 if x_t is a stop word or is inside the answer phrase. In abstractive summarization, we generate focus guide using copy target generation by Gehrmann et al. (2018), where they set source document token x_t is *copied* if it is part of the longest possible subsequence that overlaps with the target summary.

Alg. 1 describes the overall training process, which first uses stochastic hard-EM (Neal and Hinton, 1998; Lee et al., 2016) for training the SELECTOR and then the canonical MLE for training the generator.

E-step (line 2-5 in Alg. 1) we sample focus from all experts and compute their losses $-\log p_\phi(\mathbf{m}^{\text{guide}}|\mathbf{x}, z)$. Then we choose an expert z^{best} with minimum loss.

$$z^{\text{best}} = \underset{z}{\text{argmin}} -\log p_\phi(\mathbf{m}^{\text{guide}}|\mathbf{x}, z) \quad (4)$$

M-step (line 6 in Alg. 1) we only update the parameters of the chosen expert z^{best}.

Training Generator (line 7-8 in Alg. 1) The generator is independently trained using conventional teacher forcing, which minimizes $-\log p_\theta(\mathbf{y}|\mathbf{x}, \mathbf{m}^{\text{guide}})$.

4 Experimental Setup

We describe our experimental setup for question generation (Section 4.1) and abstractive summarization (Section 4.2). For both tasks, we use an off-the-shelf, task-specific encoder-decoder-based model for the generator and show how adding SELECTOR can help to diversify the output of an arbitrary generator. To evaluate the contribution of SELECTOR we additionally compare our method with previous diversity-promoting methods as the baseline (Section 4.3).

4.1 Question Generation

Question generation is the task of generating a question from a passage-answer pair. Answers are given as a span in the passage (See Fig.1).

Dataset We conduct experiments on SQuAD (Rajpurkar et al., 2016) and use the same dataset split of Zhou et al. (2017a), resulting in 86,635, 8,965, and 8,964 source-target pairs for training, validation, and test, respectively. Both source passages and target questions are single sentences. The average length of source passage and target question are 32 and 11 tokens.

Generator We use NQG++ (Zhou et al., 2017a) as the generator, which is an RNN-based encoder-decoder architecture with copying mechanism (Gulcehre et al., 2016).

4.2 Abstractive Summarization

Abstractive summarization is the task of generating a summary sentence from a source document that consists of multiple sentences.

Dataset We conduct experiments on the non-anonymized version of CNN-DM dataset (Hermann et al., 2015; Nallapati et al., 2016; See et al., 2017), whose training, validation, test splits have size of 287,113, 13,368, and 11,490 source-target pairs, respectively. The average length of the source documents and target summaries are 386 and 55 tokens. Following See et al. (2017), we truncate source and target sentences to 400 and 100 tokens during training.

Generator We use Pointer Generator (PG) (See et al., 2017) as the generator for summarization, which also leverages RNN-based encoder-decoder architecture and copying mechanism, and uses coverage loss to avoid repetitive phrases.

4.3 Baselines

For each task, we compare our method with other techniques which promote diversity at the decoding step. In particular, we compare with recent diverse search algorithms including Truncated Sampling (Fan et al., 2018), Diverse Beam Search (Vijayakumar et al., 2018), and Mixture Decoder (Shen et al., 2019). We implement these methods with NQG++ and PG. For each method, we generate K = (3 and 5) hypotheses from each source sequence. For search-based baselines (Fan et al., 2018; Vijayakumar et al., 2018), we select the top-k candidates after generation. For mixture models (Shen et al. (2019) and ours), we conduct greedy decoding from each mixture for fair comparison with search-based methods in terms of speed/memory usage.

Beam Search This baseline keeps K hypotheses with highest log-probability scores at each decoding step.

Diverse Beam Search This baseline adds a diversity promoting term to log-probability when scoring hypotheses in beam search, Following Vijayakumar et al. (2018), we use hamming diversity and diversity strength $\lambda = 0.5$.

Truncated Sampling This baseline randomly samples words from top-10 candidates of the distribution at the decoding step (Fan et al., 2018).

Mixture Decoder This baseline constructs a hard-MoE of K decoders with uniform mixing coefficient (referred as hMup in Shen et al. (2019)) and conducts parallel greedy decoding. All decoders share all parameters but use different embeddings for start-of-sequence token.

Mixture Selector (Ours) We construct a hard-MoE of K SELECTORs with uniform mixing coefficient that infers K different focus from source sequence. Guided by K focus, generator conducts parallel greedy decoding.

4.4 Metrics: Accuracy and Diversity

We use metrics introduced by previous works (Ott et al., 2018; Vijayakumar et al., 2018; Zhu et al., 2018) to evaluate the diversity promoting approaches. These metrics are extensions over BLEU-4 (Papineni et al., 2002) and ROUGE-2 F_1-score (Lin, 2004) and aim to evaluate the trade-off between accuracy and diversity.

Top-1 metric (⇑) This measures the Top-1 accuracy among the generated K-best hypotheses. The accuracy is measured using a corpus-level metric, i.e., BLEU-4 or ROUGE-2.

Oracle metric (⇑) This measures the quality of the target distribution coverage among the Top-K generated target sequences (Ott et al., 2018; Vijayakumar et al., 2018). Given an optimal ranking method (oracle), this metric measures the upper bound of Top-1 accuracy by comparing the best hypothesis with the target. Concretely, we generate hypotheses $\{\hat{\mathbf{y}}^1 \dots \hat{\mathbf{y}}^K\}$ from each source $\mathbf{x}$ and keep the hypothesis $\hat{\mathbf{y}}^{best}$ that achieves the best sentence-level metric with the target $\mathbf{y}$. Then we calculate a corpus-level metric with the greedily-selected hypotheses $\{\hat{\mathbf{y}}^{(i),best}\}_{i=1}^N$ and references $\{\mathbf{y}^{(i)}\}_{i=1}^N$.

Pairwise metric (⇓) Referred as self- (Zhu et al., 2018) or pairwise- (Ott et al., 2018) metric, this measures the within-distribution similarity. This metric computes the average of sentence-level metrics between all pairwise combinations of hypotheses $\{\hat{\mathbf{y}}^1 \dots \hat{\mathbf{y}}^K\}$ generated from each source sequence $\mathbf{x}$. Low pairwise metric indicates high diversity between generated hypotheses.

4.5 Human Evaluation Setup

We ask Amazon Mechanical Turkers (AMT) to compare our method with the baselines. For each method, we generate three questions / summaries from 100 passages sampled from SQuAD / CNN-DM test set. For every pair of methods, annotators are instructed to pick a set of questions / summaries that are more *diverse*. To evaluate *accuracy*, they see one question / summary selected

out of 3 questions / summaries with highest log-probability from each method. They are instructed to select a question / summary that is more coherent with the source passage / document. Each annotator is asked to choose either a better method (resulting in "win" or "lose") or "tie" if their quality is indistinguishable. Diversity and accuracy evaluations are conducted separately, and every pair of methods are presented to 10 annotators[1].

4.6 Implementation details

For all experiments, we tie the weights (Press and Wolf, 2017) of the encoder embedding, the decoder embedding, and the decoder output layers. This significantly reduces the number of parameters and training time until convergence. We train up to 20 epochs and select the checkpoint with the best oracle metric. We use Adam (Kingma and Ba, 2015) optimizer with learning rate 0.001 and momentum parmeters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Minibatch size is 64 and 32 for question generation and abstractive summarization. All models are implemented in PyTorch (Paszke et al., 2017) and trained on single Tesla P40 GPU, based on NAVER Smart Machine Learning (NSML) platform (Kim et al., 2018a).

Question Generation Following Zhou et al. (2017a), we use 256-dim hidden states for each direction of Bi-GRU encoder, 512-dim hidden states for GRU decoder, 300-dim word embedding initialized from GloVe (Pennington et al., 2014), vocabulary of 20,000 most frequent words, 16-dim embeddings for three linguistic features (POS, NER and word case) respectively.

Abstractive Summarization Following See et al. (2017), we use 256-dim hidden states for each direction of Bi-LSTM encoder and LSTM decoder, 128-dim word embedding trained from scratch, and vocabulary of 50,000 most frequent words. Following See et al. (2017), we train our model to generate concatenation of target summaries and split it with periods.

SELECTOR The GRU has the same size as the generator encoder, and the dimension of expert embedding $\mathbf{e}_z$ is 300 for NQG++, and 128 for PG. From simple grid search over [0.1, 0.5], we obtain focus binarization threshold th 0.15. The size of focus embedding for the generator is 16.

[1]We conducted 12 human evaluations in total: 2 tasks x 3 baselines x 2 criteria. See Table 3a and 3b.

Method	BLEU-4 (Top-1)	Oracle (Top-K)	Pairwise (Self-sim)
NQG++	13.27	-	-
Search-based Methods			
3-Beam	13.590	16.848	67.277
5-Beam	13.526	18.809	74.674
3-D. Beam	13.696	16.989	68.018
5-D. Beam	13.379	18.298	74.795
3-T. Sampling	11.890	15.447	**37.372**
5-T. Sampling	11.530	17.651	45.990
Mixture of Experts + Greedy Decoding			
3-M. Decoder	14.720	19.324	51.360
5-M. Decoder	15.166	21.965	58.727
3-M. SELECTOR (Ours)	**15.874**	20.437	47.493
5-M. SELECTOR (Ours)	15.672	**22.451**	59.815
Focus Guide during Test Time			
5-Beam + Focus Guide	24.580		-

Table 1: **Question generation results:** Comparison of diverse generation methods on SQuAD. The score of NQG++ (top row) is from Zhou et al. (2017a), and the rest are from our experiments using NQG++ as a generator. Method prefixes are the numbers of generated questions for each passage-answer pair. Best scores are bolded.

5 Results

Diversity vs. Accuracy Trade-off Tables 1 and 2 compare our method with different diversity-promoting techniques in question generation and abstractive summarization. The tables show that our mixture SELECTOR method outperforms all baselines in Top-1 and oracle metrics and achieves the best trade-off between diversity and accuracy. Moreover, both mixture models are superior to all search-based methods in the trade-off between diversity and accuracy. Standard and diverse beam search methods score low both in terms of accuracy and diversity. Truncated sampling shows the lowest self-similarity (high diversity), but it achieves the lowest score on Top-1 accuracy. Notably, our method scores state-of-the-art BLEU-4 in question generation on SQuAD and ROUGE comparable to state-of-the-art methods in abstractive summarization in CNN-DM (See also Table 4 for state-of-the-art results in CNN-DM).

Diversity vs. Number of Mixtures Here we compare the effect of number of mixtures in our SELECTOR and Mixture Decoder (Shen et al., 2019). Tables 1 and 2 show that pairwise similarity increases (diversity ⇓) when the number of mixtures increases for Mixture Decoder. While we observe a similar trend for SELECTOR in the question generation task, Table 2 shows the opposite

Method	ROUGE-2 (Top-1)	Oracle (Top-K)	Pairwise (Self-sim)
PG	17.28	-	-
Search-based Methods			
3-Beam	16.533	18.509	85.598
5-Beam	16.634	19.442	84.765
3-D. Beam	16.667	18.722	85.496
5-D. Beam	16.632	19.659	84.043
3-T. Sampling	12.914	17.068	17.306
5-T. Sampling	13.049	19.161	**16.720**
Mixture of Experts + Greedy Decoding			
3-M. Decoder	15.854	21.214	43.168
5-M. Decoder	16.104	21.801	67.196
3-M. SELECTOR (Ours)	17.930	21.316	51.092
5-M. SELECTOR (Ours)	**18.309**	**22.511**	47.280
Focus Guide during Test Time			
5-Beam + Focus Guide	42.757		-

Table 2: **Summarization results:** Comparison of diverse generation methods on CNN-DM. The score of PG (top row) is from See et al. (2017), and the rest are from our experiments using PG as a generator. Method prefixes are the numbers of generated summaries for each document. Best scores are bolded.

	Diversity (%)			Accuracy (%)		
Baselines	Win	Lose	Tie	Win	Lose	Tie
vs. 3-D. Beam	**49.7**	31.3	19.0	**43.9**	36.9	19.2
vs. 3-T. Sampling	**46.7**	35.1	18.2	**45.3**	36.1	18.6
vs. 3-M. Decoder	**47.6**	32.5	19.9	**41.8**	36.0	22.2

(a) SQuAD question generation

	Diversity (%)			Accuracy (%)		
Baselines	Win	Lose	Tie	Win	Lose	Tie
vs. 3-D. Beam	**50.4**	40.9	8.7	**46.2**	38.5	15.3
vs. 3-T. Sampling	**48.7**	42.0	9.3	**50.3**	41.2	8.5
vs. 3-M. Decoder	**49.7**	39.6	10.7	**46.5**	37.5	16.0

(b) CNN-DM abstractive summarization

Table 3: Human evaluation results

effect in the summarization task i.e., pairwise similarity decreases (diversity ⇑) for SELECTOR.

The abstractive summarization task on CNN-DM has a target distribution with more modalities than question generation task on SQuAD, which is more difficult to model. We speculate that our SELECTOR improves accuracy by focusing on more modes of the output distribution (diversity ⇑), whereas Mixture Decoder tries to improve the accuracy by concentrating on fewer modalities of the output distribution (diversity ⇓).

Upper Bound Performance The bottom rows of Tables 1 and 2 show the upper bound performance of SELECTOR by feeding focus guide to generator during test time. In particular, we assume that the mask is the ground truth overlap between input and target sequences at test time. The gap between the oracle metric (top-k accuracy) and the upper bound is very small for question generation. This indicates that the top-k masks for question generation include the ground truth mask. Future work involves improving the content selection stage for the summarization task.

Human Evaluation Table 3a and 3b show the human evaluation in two tasks of questions generation and summarization, comparing sequences generated by SELECTOR with the diversity-promoting baselines: Diverse Beam, Truncated Sampling and Mixture Decoder. The table shows that our method significantly outperforms all three baselines in terms of both diversity and accuracy with statistical significance.

Comparison with State-of-the-art Table 4 compares the performance of SELECTOR with the state-of-the-art bottom-up content selection of Gehrmann et al. (2018) in abstractive summarization. SELECTOR passes focus embeddings at the decoding step, whereas the bottom-up selection method only uses the masked words for the copy mechanism. We set K, the number of mixtures of SELECTOR, to 1 to directly compare it with the previous work (Bottom-Up (Gehrmann et al., 2018)). We observe that SELECTOR not only outperforms Bottom-Up in every metric, but also achieves a new state-of-the-art ROUGE-1 and ROUGE-L on CNN-DM. Moreover, our method scores state-of-the-art BLEU-4 in question generation on SQuAD (Table 1).

Method	R-1	R-2	R-L
PG (See et al., 2017)	39.53	17.28	36.38
Bottom-Up (Gehrmann et al., 2018)	41.22	18.68	38.34
DCA (Celikyilmaz et al., 2018)	41.69	**19.47**	37.92
SELECTOR & 10-Beam PG (Ours)	**41.72**	18.74	**38.79**

Table 4: Comparison of single-expert selector with state-of-the-art abstractive summarization methods on CNN-DM. R stands for ROUGE (Lin, 2004)

Efficient Training Table 5 shows that SELECTOR trains up to 3.7 times faster than mixture decoder (Shen et al., 2019). Training time of mixture decoder linearly increases with the number of decoders, while parallel focus inference of SELECTOR makes additional training time negligible.

Qualitative Analysis To analyze how the generator uses the selected content, we visualize attention heatmap of NQG++ in Fig.3 for question generation. The figure shows different attention

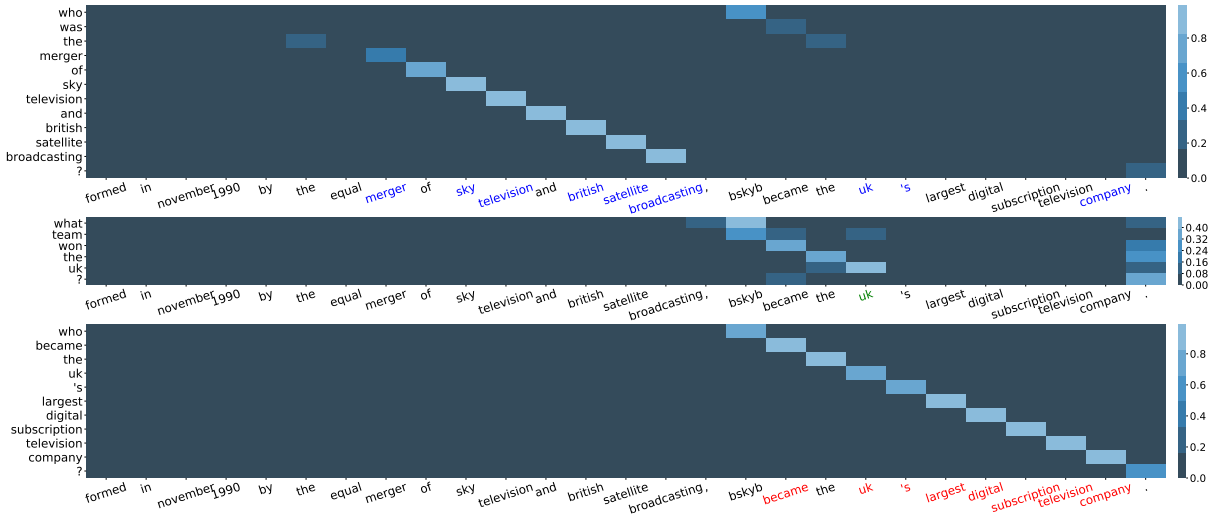

Figure 3: Attention heatmap of NQG++ decoder with three different focus by SELECTOR. This shows focus can guide generator to generate different sequences. Passage tokens are colored when corresponding focus is 1.

Method	Training time (ms. / step)
PG	641.2
3-M. Decoder	1804.1 (× 2.81)
5-M. Decoder	2367.6 (× 4.37)
SELECTOR (Ours)	692.1 (× 1.08)
3-M. SELECTOR (Ours)	740.8 (× 1.16)
5-M. SELECTOR (Ours)	747.6 (× 1.17)

Table 5: **Training time:** Comparison of training time on CNN-DM. See 4.6 for implementation details.

mechanisms depending on three different focuses inferred by different SELECTOR experts.

6 Conclusion

We introduce a novel diverse sequence generation method via proposing a content selection module, SELECTOR. Built upon mixture of experts and hard-EM training, SELECTOR identifies different key parts on source sequence to guide generator to output a diverse set of sequences.

SELECTOR is a generic plug-and-play module that can be added to an existing encoder-decoder model to enforce diversity with a negligible additional computational cost. We empirically demonstrate that our method improves both accuracy and diversity and reduces training time significantly compared to baselines in question generation and abstractive summarization. Future work involves incorporating SELECTOR for other generation tasks such as diverse image captioning.

Acknowledgment

This research was supported by Allen Distinguished Investigator Award, the Office of Naval Research under the MURI grant N00014-18-1-2670, Samsung GRO, and gifts from Allen Institute for AI and Google. We also thank the members of Clova AI, UW NLP, and the anonymous reviewers for their insightful comments.

References

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating Sentences from a Continuous Space. In *CoNLL*.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep Communicating Agents for Abstractive Summarization. In *NAACL*.

Kyunghyun Cho. 2016. Noisy Parallel Approximate Decoding for Conditional Recurrent Language Model.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*.

Jan Chorowski and Navdeep Jaitly. 2017. Towards better decoding and language model integration in sequence to sequence models. In *INTERSPEECH*.

Yuntian Deng, Yoon Kim, Justin Chiu, Demi Guo, and Alexander M. Rush. 2018. Latent Alignment and Variational Attention. In *NIPS*.

Adji B Dieng, Yoon Kim, Alexander M Rush, and David M Blei. 2018. Avoiding Latent Variable Collapse with Generative Skip Models. In *ICML Workshop*.

David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. 2014. Learning Factored Representations in a Deep Mixture of Experts. In *ICLR Workshop*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical Neural Story Generation. In *ACL*.

Sebastian Gehrmann, Yuntian Deng, and Alexander M Rush. 2018. Bottom-Up Abstractive Summarization. In *EMNLP*.

Alex Graves. 2013. Generating Sequences With Recurrent Neural Networks.

Xiaodong Gu, Kyunghyun Cho, Jung-woo Ha, and Sunghun Kim. 2019. DialogWAE : Multimodal Response Generation with Conditional Wasserstein Auto-Encoder. In *ICLR*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the Unknown Words. In *ACL*.

Abner Guzman-Rivera, Dhruv Batra, and Pushmeet Kohli. 2012. Multiple Choice Learning: Learning to Produce Multiple Structured Outputs. In *NIPS*.

Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. 2019. Lagging Inference Networks and Posterior Collapse in Variational Autoencoders. In *ICLR*.

Xuanli He, Gholamreza Haffari, and Mohammad Norouzi. 2018. Sequence to Sequence Mixture Model for Diverse Machine Translation. In *CoNLL*.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In *NIPS*.

Robert Jacobs, Michael Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. Adaptive Mixtures of Local Experts. *Neural Computation*, 3(1):78–87.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *ICLR*.

Ashwin Kalyan, Stefan Lee, Anitha Kannan, and Dhruv Batra. 2018. Learn from Your Neighbor: Learning Multi-modal Mappings from Sparse Annotations. In *ICML*.

Nan Rosemary Ke, Konrad Zolna, Alessandro Sordoni, Zhouhan Lin, Adam Trischler, Yoshua Bengio, Joelle Pineau, Laurent Charlin, and Chris Pal. 2018. Focused Hierarchical RNNs for Conditional Sequence Processing. In *ICML*.

Hanjoo Kim, Minkyu Kim, Dongjoo Seo, Jinwoong Kim, Heungseok Park, Soeun Park, Hyunwoo Jo, KyungHyun Kim, Youngil Yang, Youngkwan Kim, Nako Sung, and Jung-Woo Ha. 2018a. NSML: Meet the MLaaS platform with a real-world case study.

Yoon Kim, Sam Wiseman, Andrew C. Miller, David Sontag, and Alexander M. Rush. 2018b. Semi-Amortized Variational Autoencoders. In *ICML*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

Diederik P Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. In *NIPS*.

Stefan Lee, Senthil Purushwalkam, Michael Cogswell, Viresh Ranjan, David Crandall, and Dhruv Batra. 2016. Stochastic Multiple Choice Learning for Training Diverse Deep Ensembles. In *NIPS*.

Chenliang Li, Weiran Xu, Si Li, and Sheng Gao. 2018. Guiding Generation for Abstractive Text Summarization based on Key Information Guide Network. In *NAACL*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and William B. Dolan. 2016a. A Diversity-Promoting Objective Function for Neural Conversation Models. In *NAACL*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016b. A Simple, Fast Diverse Decoding Algorithm for Neural Generation.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *ACL Workshop*.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *ICLR*.

Sewon Min, Victor Zhong, Richard Socher, and Caiming Xiong. 2018. Efficient and Robust Question Answering from Minimal Context over Documents. In *ACL*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. In *CoNLL*.

Radford M. Neal and Geoffrey E. Hinton. 1998. A View of the EM Algorithm that Justifies Incremental, Sparse, and other Variants. In *Learning in Graphical Models*, pages 355–368.

Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Analyzing Uncertainty in Neural Machine Translation. In *ICML*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wj Wei-jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *ACL*.

Yookoon Park, Jaemin Cho, and Gunhee Kim. 2018. A Hierarchical Latent Structure for Variational Conversation Modeling. In *NAACL*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chana, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS Workshop*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*.

Ofir Press and Lior Wolf. 2017. Using the Output Embedding to Improve Language Models. In *EACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *EMNLP*.

Ali Razavi, Aaron van den Oord, Ben Poole, and Oriol Vinyals. 2019. Preventing Posterior Collapse with δ-VAEs. In *ICLR*.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *ACL*.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues. In *AAAI*.

Shiv Shankar and Sunita Sarawagi. 2019. Posterior Attention Models for Sequence to Sequence Learning. In *ICLR*.

Tianxiao Shen, Myle Ott, Michael Auli, and Marc'Aurelio Ranzato. 2019. Mixture Models for Diverse Machine Translation: Tricks of the Trade. In *ICML*.

Jinsong Su, Shan Wu, Deyi Xiong, Yaojie Lu, Xianpei Han, and Biao Zhang. 2018. Variational Recurrent Neural Machine Translation. In *AAAI*.

Sandeep Subramanian, Tong Wang, Xingdi Yuan, Saizheng Zhang, Yoshua Bengio, and Adam Trischler. 2018. Neural Models for Key Phrase Extraction and Question Generation. In *ACL Workshop*.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath Ramprasaath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse Beam Search for Improved Description of Complex Scenes. In *AAAI*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer Networks. In *NIPS*.

Tsung-hsien Wen and Minh-thang Luong. 2018. Latent Topic Conversational Models.

Tsung Hsien Wen, Yishu Miao, Phil Blunsom, and Steve Young. 2017. Latent Intention Dialogue Models. In *ICML*.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256.

Jiacheng Xu and Greg Durrett. 2018. Spherical Latent Spaces for Stable Variational Autoencoders. In *EMNLP*.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the Softmax Bottleneck: A High-Rank RNN Language Model. In *ICLR*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. In *AAAI*.

Biao Zhang, Deyi Xiong, Jinsong Su, Hong Duan, and Min Zhang. 2016. Variational Neural Machine Translation. In *EMNLP*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning Discourse-level Diversity for Neural Dialog Models using Conditional Variational Autoencoders. In *ACL*.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017a. Neural Question Generation from Text: A Preliminary Study. In *NLPCC*.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017b. Selective Encoding for Abstractive Sentence Summarization. In *ACL*.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A Benchmarking Platform for Text Generation Models. In *SIGIR*.