QUESTION GENERATION FROM PARAGRAPHS: A TALE OF TWO HIERARCHICAL MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Automatic question generation from paragraphs is an important and challenging problem, particularly due to the long context from paragraphs. In this paper, we propose and study two hierarchical models for the task of question generation from paragraphs. Specifically, we propose (a) a novel hierarchical BiLSTM model with selective attention and (b) a novel hierarchical Transformer architecture, both of which learn hierarchical representations of paragraphs. We model a paragraph in terms of its constituent sentences, and a sentence in terms of its constituent words. While the introduction of the attention mechanism benefits the hierarchical BiLSTM model, the hierarchical Transformer, with its inherent attention and positional encoding mechanisms also performs better than flat transformer model. We conducted empirical evaluation on the widely used SQuAD and MS MARCO datasets using standard metrics. The results demonstrate the overall effectiveness of the hierarchical models over their flat counterparts. Qualitatively, our hierarchical models are able to generate fluent and relevant questions.

1 INTRODUCTION

Question Generation (QG) from text has gained significant popularity in recent years in both academia and industry, owing to its wide applicability in a range of scenarios including conversational agents, automating reading comprehension assessment, and improving question answering systems by generating additional training data. Neural network based methods represent the state-of-the-art for automatic question generation. These models do not require templates or rules, and are able to generate fluent, high-quality questions.

Most of the work in question generation takes sentences as input (Du & Cardie, 2018; Kumar et al., 2018; Song et al., 2018; Kumar et al., 2019). QG at the paragraph level is much less explored and it has remained a challenging problem. The main challenges in paragraph-level QG stem from the larger context that the model needs to assimilate in order to generate relevant questions of high quality.

Existing question generation methods are typically based on recurrent neural networks (RNN), such as bi-directional LSTM. Equipped with different enhancements such as the attention, copy and coverage mechanisms, RNN-based models (Du et al., 2017; Kumar et al., 2018; Song et al., 2018) achieve good results on sentence-level question generation. However, due to their ineffectiveness in dealing with long sequences, paragraph-level question generation remains a challenging problem for these models.

Recently, Zhao et al. (2018) proposed a paragraph-level QG model with maxout pointers and a gated self-attention encoder. To the best of our knowledge this is the only model that is designed to support paragraph-level QG and outperforms other models on the SQuAD dataset (Rajpurkar et al., 2016). One straightforward extension to such a model would be to reflect the structure of a paragraph in the design of the encoder. Our first attempt is indeed a hierarchical BiLSTM-based paragraph encoder (HPE), wherein, the hierarchy comprises the word-level encoder that feeds its encoding to the sentence-level encoder. Further, dynamic paragraph-level contextual information in the BiLSTM-HPE is incorporated via both word- and sentence-level selective attention.

However, LSTM is based on the recurrent architecture of RNNs, making the model somewhat rigid and less dynamically sensitive to different parts of the given sequence. Also LSTM models are

slower to train. In our case, a paragraph is a sequence of sentences and a sentence is a sequence of words. The Transformer (Vaswani et al., 2017) is a recently proposed neural architecture designed to address some deficiencies of RNNs. Specifically, the Transformer is based on the (multi-head) attention mechanism, completely discarding recurrence in RNNs. This design choice allows the Transformer to effectively attend to different parts of a given sequence. Also Transformer is relatively much faster to train and test than RNNs.

As humans, when reading a paragraph, we look for important sentences first and then important keywords in those sentences to find a concept around which a question can be generated. Taking this inspiration, we give the same power to our model by incorporating word-level and sentence-level selective attention to generate high-quality questions from paragraphs.

In this paper, we present and contrast novel approachs to QG at the level of paragraphs. Our main contributions are as follows:

- We present two hierarchical models for encoding the paragraph based on its structure. We analyse the effectiveness of these models for the task of automatic question generation from paragraph.
- Specifically, we propose a novel hierarchical Transformer architecture. At the lower level, the encoder first encodes words and produces a sentence-level representation. At the higher level, the encoder aggregates the sentence-level representations and learns a paragraph-level representation.
- We also propose a novel hierarchical BiLSTM model with selective attention, which learns to attend to important sentences and words from the paragraph that are relevant to generate meaningful and fluent questions about the encoded answer.
- We also present attention mechanisms for dynamically incorporating contextual information in the hierarchical paragraph encoders and experimentally validate their effectiveness.

2 Related Work

Question generation (QG) has recently attracted significant interests in the natural language processing (NLP) (Du et al., 2017; Kumar et al., 2018; Song et al., 2018; Kumar et al., 2019) and computer vision (CV) (Li et al., 2018; Fan et al., 2018) communities. Given an input (*e.g.*, a passage of text in NLP or an image in CV), optionally also an answer, the task of QG is to generate a natural-language question that is answerable from the input.

Existing text-based QG methods can be broadly classified into three categories: (a) rule-based methods, (b) template-base methods, and (c) neural network-based methods. Rule based methods (Heilman & Smith, 2010) perform syntactic and semantic analysis of sentences and apply fixed sets of rules to generate questions. They mostly rely on syntactic rules written by humans (Heilman, 2011) and these rules change from domain to domain. On the other hand, template based methods (Ali et al., 2010) use generic templates/slot fillers to generate questions. More recently, neural network-based QG methods (Du et al., 2017; Kumar et al., 2018; Song et al., 2018) have been proposed. They employ an RNN-based encoder-decoder architecture and train in an end-to-end fashion, without the need of manually created rules or templates.

Du et al. (2017) were the first to propose a sequence-to-sequence (Seq2Seq) architecture for QG. Kumar et al. (2018) proposed to augment each word with linguistic features and encode the most relevant *pivotal answer* in the text while generating questions. Similarly, Song et al. (2018) encode ground-truth answers (given in the training data), use the copy mechanism and additionally employ context matching to capture interactions between the answer and its context within the passage. They encode ground-truth answer for generating questions which might not be available for the test set.

Zhao et al. (2018) recently proposed a Seq2Seq model for paragraph-level question generation, where they employ a maxout pointer mechanism with a gated self-attention encoder. Tran et al. (2018) contrast recurrent and non-recurrent architectures on their effectiveness in capturing the hierarchical structure. In Machine Translation, non-recurrent model such as a Transformer (Vaswani et al., 2017) that does not use convolution or recurrent connection is often expected to perform better. However, Transformer, as a non-recurrent model, can be more effective than the recurrent model

because it has full access to the sequence history. Our findings also suggest that LSTM outperforms the Transformer in capturing the hierarchical structure. In contrast, Goldberg (2019) report settings in which attention-based models, such as BERT are better capable of learning hierarchical structure than LSTM-based models.

3 HIERARCHICAL PARAGRAPH REPRESENTATION

We propose a general hierarchical architecture for better paragraph representation at the level of words and sentences. This architecture is agnostic to the type of encoder, so we base our hierarchical architectures on BiLSTM and Transformers. We then present two decoders (LSTM and Transformer) with hierarchical attention over the paragraph representation, in order to provide the dynamic context needed by the decoder. The decoder is further conditioned on the provided (candidate) answer to generate relevant questions.

Notation: The question generation task consists of pairs $(\boldsymbol{X}, \boldsymbol{y})$ conditioned on an encoded answer $\boldsymbol{z}$, where $\boldsymbol{X}$ is a paragraph, and $\boldsymbol{y}$ is the target question which needs to be generated with respect to the paragraph.. Let us denote the i-th sentence in the paragraph by $\boldsymbol{x}_i$, where $x_{i,j}$ denotes the j-th word of the sentence. We assume that the first and last words of the sentence are special beginning-of-the-sentence $<$ BOS $>$ and end-of-the-sentence $<$ EOS $>$ tokens, respectively.

3.1 HIERARCHICAL PARAGRAPH ENCODER

Our hierarchical paragraph encoder (HPE) consists of two encoders, *viz.*, a sentence-level and a word-level encoder; (*c.f.* Figure 1).

Word-Level Encoder: The lower-level encoder WORDENC encodes the words of individual sentences. This encoder produces a sentence-dependent word representation $\boldsymbol{r}_{i,j}$ for each word $x_{i,j}$ in a sentence $\boldsymbol{x}_i$, *i.e.*, $\boldsymbol{r}_i = \text{WORDENC}\ (\boldsymbol{x}_i)$. This representation is the output of the last encoder block in the case of Transformer, and the last hidden state in the case of BiLSTM. Furthermore, we can produce a fixed-dimensional representation for a sentence as a function of $\boldsymbol{r_i}$, *e.g.*, by summing (or averaging) its contextual word representations, or concatenating the contextual representations of its $<$ BOS $>$ and $<$ EOS $>$ tokens. We denote the resulting sentence representation by $\tilde{\boldsymbol{s}}_i$ for a sentence $\boldsymbol{x}_i$.

Sentence-Level Encoder: At the higher level, our HPE consists of another encoder to produce paragraph-dependent representation for the sentences. The input to this encoder are the sentence representations produced by the lower level encoder, which are insensitive to the paragraph context. In the case of the transformer, the sentence representation is combined with its positional embedding to take the ordering of the paragraph sentences into account. The output of the higher-level encoder is contextual representation for each set of sentences $\boldsymbol{s} = \text{SENTENC}\ (\tilde{\boldsymbol{s}})$, where $\boldsymbol{s}_i$ is the paragraph-dependent representation for the i-th sentence.

In the following two sub-sections, we present our two hierarchical encoding architectures, *viz.*, the hierarchical BiLSTM in Section 3.2) and hierarchical transformer in Section 3.3).

3.2 DYNAMIC CONTEXT IN BILSTM-HPE

In this first option, *c.f.*, Figure 1, we use both word-level attention and sentence level attention in a Hierarchical BiLSTM encoder to obtain the hierarchical paragraph representation. We employ the attention mechanism proposed in (Luong et al., 2015) at both the word and sentence levels. We employ the BiLSTM (Bidirectional LSTM) as both, the word as well as the sentence level encoders. We concatenate forward and backward hidden states to obtain sentence/paragraph representations. Subsequently, we employ a unidirectional LSTM unit as our decoder, that generates the target question one word at a time, conditioned on (i) all the words generated in the previous time steps and (ii) on the encoded answer. The methodology employed in these modules has been described next.

Word-level Attention: We use the LSTM decoder's previous hidden state and the word encoder's hidden state to compute attention over words (Figure 1). We the concatenate forward and backward

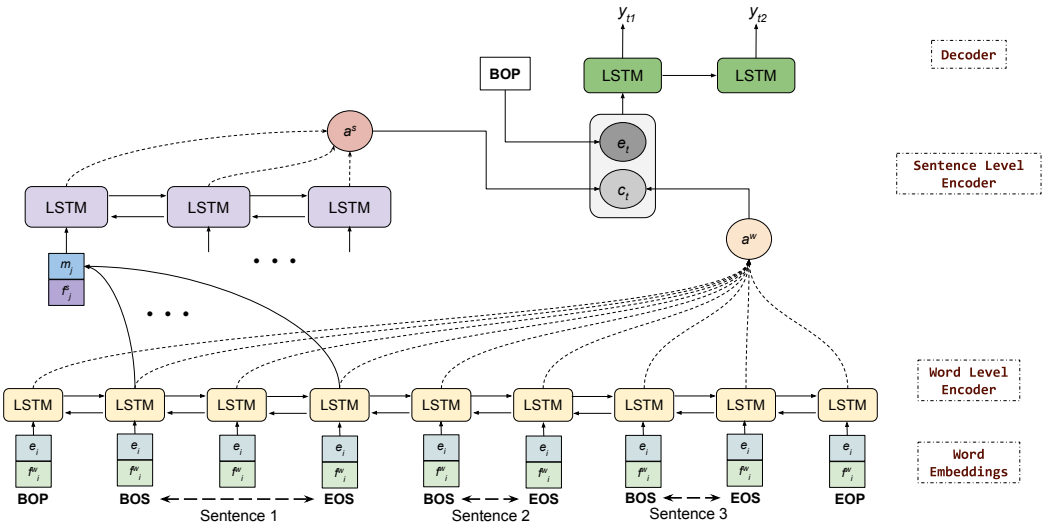

Figure 1: Our hierarchial LSTM Architecture.

hidden states of the BiLSTM encoder to obtain the final hidden state representation (h_t) at time step t. Representation (h_t) is calculated as: $\boldsymbol{h}_t = \text{WORDENC}\ (\mathbf{h}_{t-1}, [\boldsymbol{e}_t, \boldsymbol{f}_t^w])$, where $\boldsymbol{e}_t$ represents the GLoVE (Pennington et al., 2014) embedded representation of word ($x_{i,j}$) at time step t and $\boldsymbol{f}_t^w$ is the embedded BIO feature for answer encoding.

The word level attention ($\boldsymbol{a}_t^w$) is computed as: $\boldsymbol{a}_t^w = Softmax([u_{t_i}^w]_{i=1}^M)$, where M is the number of words, and $u_{t_i}^w = \boldsymbol{v}_w^T \tanh(\mathbf{W}_w[\boldsymbol{h}_i, \boldsymbol{d}_t])$ and $\boldsymbol{d}_t$ is the decoder hidden state at time step t.

We calculate sentence representation ($\tilde{\boldsymbol{s}}_i$) using word level encoder's hidden states as: $\tilde{\boldsymbol{s}}_i = \frac{1}{|\boldsymbol{x}_i|}\sum_j \boldsymbol{r}_{i,j}$, where $\boldsymbol{r}_{i,j}$ is the word encoder hidden state representation of the j^{th} word of the i^{th} sentence.

Sentence-level Attention: We feed the sentence representations $\tilde{\boldsymbol{s}}$ to our sentence-level BiLSTM encoder (*c.f.* Figure 1). Similar to the word-level attention, we again the compute attention weight over every sentence in the input passage, using (i) the previous decoder hidden state and (ii) the sentence encoder's hidden state. As before, we concatenate the forward and backward hidden states of the sentence level encoder to obtain the final hidden state representation. The hidden state ($\boldsymbol{g}_t$) of the sentence level encoder is computed as: $\boldsymbol{g}_t = \text{SENTENC}\ (\boldsymbol{g}_{t-1}, [\tilde{\boldsymbol{s}}_t, \boldsymbol{f}_t^s])$, where $\boldsymbol{f}_t^s$ is the embedded feature vector denoting whether the sentence contains the encoded answer or not.

The selective sentence level attention ($\boldsymbol{a}_t^s$) is computed as: $\boldsymbol{a}_t^s = Sparsemax([u_{t_i}^w]_{i=1}^K)$, where, K is the number of sentences, $u_{t_i}^s = \boldsymbol{v}_s^T \tanh(\mathbf{W}_s[\boldsymbol{g}_i, \boldsymbol{d}_t])$.

The final context ($\boldsymbol{c}_t$) based on hierarchical selective attention is computed as: $\boldsymbol{c}_t = \sum_i a_{t_i}^s \sum_j \overline{a}_{t_{i,j}}^w \boldsymbol{r}_{i,j}$, where $\overline{a}_{t_{i,j}}^w$ is the word attention score obtained from $\boldsymbol{a}_t^w$ corresponding to j^{th} word of the i^{th} sentence.

The context vector $\boldsymbol{c}_t$ is fed to the decoder at time step t along with embedded representation of the previous output.

3.3 DYNAMIC CONTEXT IN TRANSFORMER-HPE

In this second option (*c.f.* Figure 2), we make use of a Transformer decoder to generate the target question, one token at a time, from left to right. For generating the next token, the decoder attends to the previously generated tokens in the question, the encoded answer and the paragraph. We postulate that attention to the paragraph benefits from our hierarchical representation, described in Section 3.1. That is, our model identifies firstly the relevance of the sentences, and then the relevance of the words within the sentences. This results in a hierarchical attention module (HATT) and its multi-

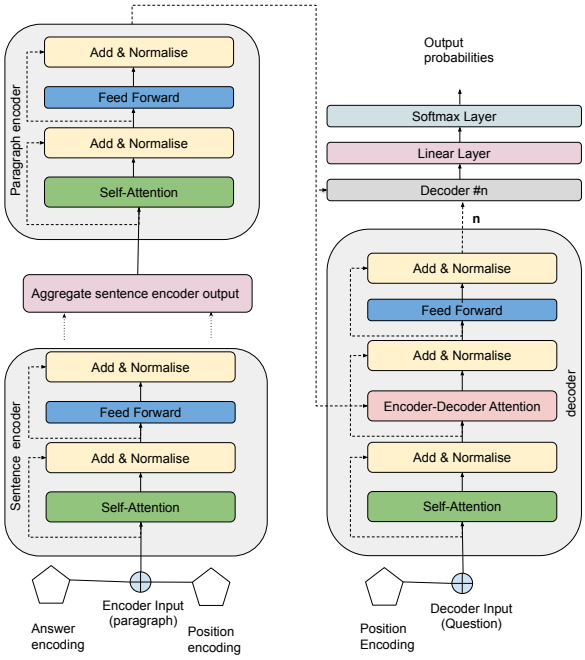

Figure 2: Our hierarchical Transformer architecture.

head extension (MHATT), which replace the attention mechanism to the source in the Transformer decoder. We first explain the sentence and paragragh encoders (Section 3.3.1) before moving on to explanation of the decoder (Section 3.3.2) and the hierarchical attention modules (HATT and MHATT in Section 3.3.3).

3.3.1 SENTENCE AND PARAGRAPH ENCODER

The sentence encoder transformer maps an input sequence of word representations $\mathbf{x} = (x_0, \cdots, x_n)$ to a sequence of continuous sentence representations $\mathbf{r} = (r_0, \cdots, r_n)$. Paragraph encoder takes concatenation of word representation of the start word and end word as input and returns paragraph representation. Each encoder layer is composed of two sub-layers namely a multi-head self attention layer (Section 3.3.3) and a position wise fully connected feed forward neural network (Section 3.3.4). To be able to effectively describe these modules, we will benefit first from a description of the decoder (Section 3.3.2).

3.3.2 DECODER

The decoder stack is similar to encoder stack except that it has an additional sub layer (encoder-decoder attention layer) which learn multi-head self attention over the output of the paragraph encoder.

The output of the paragraph encoder is transformed into a set of attention vectors K_{encdec} and V_{encdec}. Encoder-decoder attention layer of decoder takes the key K_{encdec} and value V_{encdec} . Decoder stack will output a float vector, we can feed this float vector to a linear followed softmax layer to get probability for generating target word.

3.3.3 THE HATT AND MHATT MODULES

Let us assume that the question decoder needs to attend to the source paragraph during the generation process. To attend to the hierarchical paragraph representation, we replace the multi-head attention mechanism (to the source) in Transformer by introducing a new multi-head hierarchical attention module $MHATT(q^s, K^s, q^w, K^w, V^w)$ where q^s is the sentence-level query vector, q^w is the word

level query vector, K^s is the key matrix for the sentences of the paragraph, K^w is the key matrix for the words of the paragraph, and V^w is the value matrix fr the words of the paragraph.

The vectors of sentence-level query q^s and word-level query q^s are created using non-linear transformations of the state of the decoder $\boldsymbol{h}_{t-1}$, i.e. the input vector to the softmax function when generating the previous word w_{t-1} of the question. The matrices for the sentence-level key K^s and word-level key K^w are created using the output. We take the input vector to the softmax function $\boldsymbol{h}_{t-1}$, when the t-th word in the question is being generated. Firstly, this module attends to paragraph sentences using their keys and the sentence query vector: $\boldsymbol{a} = softmax(q_s K_s/d)$. Here, d is the dimension of the query/key vectors; the dimension of the resulting attention vector would be the number of sentences in the paragraph. Secondly, it computes an attention vector for the words of each sentence: $\boldsymbol{b}_i = softmax(q_w K_w^i/d)$. Here, K_w^i is the key matrix for the words in the i-th sentences; the dimension of the resulting attention vector $\boldsymbol{b}_i$ is the number of tokens in the i-th sentence. Lastly, the context vector is computed using the word values of their attention weights based on their sentence-level and word-level attentions:

$$HATT(q_s, K_s, q_w, K_w, V_w) = \sum_{i=1}^{|\boldsymbol{d}|} a_i \big(\boldsymbol{b}_i \cdot V_i^w\big)$$

Attention in MHATT module is calculated as:

$$\textit{Attention}(Q_w, V_w, K_w) = \textit{softmax}(\frac{Q_w K_w}{\sqrt{d_k}})V_w \tag{1}$$

Where $\textit{Attention}(Q_w, V_w, K_w)$ is reformulation of scaled dot product attention of (Vaswani et al., 2017). For multiple heads, the multihead attention $\mathbf{z} = \textit{Multihead}(Q_w, K_w, V_w)$ is calculated as:

$$\textit{Multihead}(Q_w, K_w, V_w) = \textit{Concat}(h_0, h_2, .., h_n)W^O \tag{2}$$

where $h_i = \textit{Attention}(Q_w W_i^Q, K_w W_i^K, V_w W_i^V)$, $W_i^Q \in R^{d_{model} \times d_k}$, $W_i^K \in R^{d_{model} \times d_k}$, $W_i^V \in R^{d_{model} \times d_v}$, $W^O \in R^{hd_v \times d_{model}}$, $d_k = d_v = d_{model}/h = 64$.

z is fed to a position-wise fully connected feed forward neural network to obtain the final input representation.

3.3.4 Position-wise Fully Connected Feed Forward Neural Network

Output of the HATT module is passed to a fully connected feed forward neural net (FFNN) for calculating the hierarchical representation of input (**r**) as: $\mathbf{r} = FFNN(x) = (max(0, xW_1 + b1))W_2 + b_2$, where **r** is fed as input to the next encoder layers. The final representation **r** from last layer of decoder is fed to the linear followed by softmax layer for calculating output probabilities.

4 Experimental Setup

4.1 Datasets

We performed all our experiments on the publicly available SQuAD (Rajpurkar et al., 2016) and MS MARCO (Nguyen et al., 2016) datasets. SQuAD contains 536 Wikipedia articles and more than 100K questions posed about the articles by crowd-workers. We split the SQuAD train set by the ratio 90%-10% into train and dev set and take SQuAD dev set as our test set for evaluation. We take an entire paragraph in each train/test instance as input in all our experiments. MS MARCO contains passages that are retrieved from web documents and the questions are anonimized versions of BING queries. We take a subset of MS MARCO v1.1 dataset containing questions that are answerable from atleast one paragraph. We split train set as 90%-10% into train (71k) and dev (8k) and take dev set as test set (9.5k). Our split is same but our dataset also contains (para, question) tuples whose answers are not a subspan of the paragraph, thus making our task more difficult.

4.2 Evaluation metrics

For evaluating our question generation model we report the standard metrics, *viz.*, BLEU (Papineni et al., 2002) and ROUGE-L(Lin, 2004). We performed human evaluation to further analyze quality

of questions generated by all the models. We analyzed quality of questions generated on a) syntactic correctness b) semantic correctness and c) relevance to the given paragraph.

4.3 MODELS

We compare QG results of our hierarchical LSTM and hierarchical Transformer with their *flat* counterparts. We describe our models below:

Seq2Seq + att + AE is the attention-based sequence-to-sequence model with a BiLSTM encoder, answer encoding and an LSTM decoder.

HierSeq2Seq + AE is the hierarchical BiLSTM model with a BiLSTM sentence encoder, a BiLSTM paragraph encoder and an LSTM decoder conditioned on encoded answer.

TransSeq2Seq + AE is a Transformer-based sequence-to-sequence model with a Transformer encoder followed by a Transformer decoder conditioned on encoded answer.

HierTransSeq2Seq + AE is the hierarchical Transformer model with a Transformer sentence encoder, a Transformer paragraph encoder followed by a Transformer decoder conditioned on answer encoded.

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L
Seq2Seq + att + AE	52.86	29.02	17.06	10.26	38.17
TransSeq2Seq + AE	42.07	22.03	12.33	7.45	36.77
HierSeq2Seq + AE	54.36	**30.62**	**18.43**	**11.50**	38.83
HierTransSeq2Seq + AE	**54.49**	29.79	17.45	10.80	**41.13**

Table 1: Automatic evaluation results on the SQuAD dataset. For each metric, best result is **bolded**

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L
Seq2Seq + att + AE	35.90	23.52	15.15	10.10	31.05
TransSeq2Seq + AE	24.90	15.32	9.46	6.13	30.76
HierSeq2Seq + AE	**38.08**	**25.33**	**16.48**	**11.13**	**32.82**
HierTransSeq2Seq + AE	31.49	20.05	12.60	8.68	31.88

Table 2: Automatic evaluation results on the MS MARCO dataset. For each metric, best result is **bolded**

Model	Syntax		Semantics		Relevance	
	Score	Kappa	Score	Kappa	Score	Kappa
Seq2Seq + att + AE	86	0.57	79.33	0.61	70.66	0.56
TransSeq2Seq + AE	86	0.66	84	0.62	50	0.60
HierSeq2Seq + AE	80	0.49	73.33	0.54	**81.33**	0.64
HierTransSeq2Seq + AE	**90**	0.62	**85.33**	0.65	68	0.56

Table 3: Human evaluation results (column "Score") as well as inter-rater agreement (column "Kappa") for each model on the SQuAD test set. The scores are between 0-100, 0 being the worst and 100 being the best. Best results for each metric (column) are **bolded**. The three evaluation criteria are: (1) syntactically correct (Syntax), (2) semantically correct (Semantics), and (3) relevant to the text (Relevance).

5 RESULTS AND DISCUSSION

In Table 1 and Table 2 we present automatic evaluation results of all models on SQuAD and MS MARCO datasets respectively. We present human evaluation results in Table 3 and Table 4 respectively.

A number of interesting observations can be made from automatic evaluation results in Table 1 and Table 2:

Model	Syntax		Semantics		Relevance	
	Score	Kappa	Score	Kappa	Score	Kappa
Seq2Seq + att + AE	83.33	0.68	69.33	0.65	38.66	0.52
TransSeq2Seq + AE	80.66	0.73	73.33	0.55	35.33	0.47
HierSeq2Seq + AE	85.33	0.73	70.66	0.68	**51.33**	0.60
HierTransSeq2Seq + AE	**86**	0.87	**73.33**	0.65	32.66	0.60

Table 4: Human evaluation results (column "Score") as well as inter-rater agreement (column "Kappa") for each model on the MS MARCO test set. The scores are between 0-100, 0 being the worst and 100 being the best. Best results for each metric (column) are **bolded**. The three evaluation criteria are: (1) syntactically correct (Syntax), (2) semantically correct (Semantics), and (3) relevant to the text (Relevance).

- Overall, the hierarchical BiLSTM model HierSeq2Seq + AE shows the best performance, achieving best result on BLEU2–BLEU4 metrics on both SQuAD dataset, whereas the hierarchical Transformer model TransSeq2Seq + AE performs best on BLEU1 and ROUGE-L on the SQuAD dataset.
- Compared to the flat LSTM and Transformer models, their respective hierarchical counterparts always perform better on both the SQuAD and MS MARCO datasets.
- On the MS MARCO dataset, we observe the best consistent performance using the hierarchical BiLSTM models on all automatic evaluation metrics.
- On the MS MARCO dataset, the two LSTM-based models outperform the two Transformer-based models.

Interestingly, human evaluation results, as tabulated in Table 3 and Table 4, demonstrate that the hierarchical Transformer model TransSeq2Seq + AE outperforms all the other models on both datasets in both syntactic and semantic correctness. However, the hierarchical BiLSTM model HierSeq2Seq + AE achieves best, and significantly better, relevance scores on both datasets.

From the evaluation results, we can see that our proposed hierarchical models demonstrate benefits over their respective flat counterparts in a significant way. Thus, for paragraph-level question generation, the hierarchical representation of paragraphs is a worthy pursuit. Moreover, the Transformer architecture shows great potential over the more traditional RNN models such as BiLSTM as shown in human evaluation. Thus the continued investigation of hierarchical Transformer is a promising research avenue. In the Appendix, in Section B, we present several examples that illustrate the effectiveness of our Hierarchical models. In Section C of the appendix, we present some failure cases of our model, along with plausible explanations.

6 Conclusion

We proposed two hierarchical models for the challenging task of question generation from paragraphs, one of which is based on a hierarchical BiLSTM model and the other is a novel hierarchical Transformer architecture. We perform extensive experimental evaluation on the SQuAD and MS MARCO datasets using standard metrics. Results demonstrate the hierarchical representations to be overall much more effective than their flat counterparts. The hierarchical models for both Transformer and BiLSTM clearly outperforms their flat counterparts on all metrics in almost all cases. Further, our experimental results validate that hierarchical selective attention benefits the hierarchical BiLSTM model. Qualitatively, our hierarchical models also exhibit better capability of generating fluent and relevant questions.

References

Husam Ali, Yllias Chali, and Sadid A Hasan. Automation of question generation from sentences. In *Proceedings of QG2010: The Third Workshop on Question Generation*, pp. 58–67, 2010.

Xinya Du and Claire Cardie. Harvesting paragraph-level question-answer pairs from Wikipedia. In *ACL (1)*, pp. 1907–1917, 2018.

Xinya Du, Junru Shao, and Claire Cardie. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pp. 1342–1352, 2017.

Zhihao Fan, Zhongyu Wei, Piji Li, Yanyan Lan, and Xuanjing Huang. A question type driven framework to diversify visual question generation. In *IJCAI*, pp. 4048–4054, 2018.

Yoav Goldberg. Assessing bert's syntactic abilities. *CoRR*, abs/1901.05287, 2019.

Michael Heilman. *Automatic factual question generation from text*. PhD thesis, Carnegie Mellon University, 2011.

Michael Heilman and Noah A Smith. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 609–617. Association for Computational Linguistics, 2010.

Vishwajeet Kumar, Kireeti Boorla, Yogesh Meena, Ganesh Ramakrishnan, and Yuan-Fang Li. Automating reading comprehension by generating question and answer pairs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 335–348. Springer, 2018.

Vishwajeet Kumar, Ganesh Ramakrishnan, and Yuan-Fang Li. Putting the horse before the cart: A generator-evaluator framework for question generation from text. In *The SIGNLL Conference on Computational Natural Language Learning (CoNLL 2019)*, 2019.

Yikang Li, Nan Duan, Bolei Zhou, Xiao Chu, Wanli Ouyang, Xiaogang Wang, and Ming Zhou. Visual question generation as dual task of visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6116–6124, 2018.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *ACL 2004*, 2004.

Thang Luong, Hieu Quang Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *EMNLP*, 2015.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 311–318. Association for Computational Linguistics, 2002.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://www.aclweb.org/anthology/D14-1162.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, 2016.

Linfeng Song, Zhiguo Wang, Wael Hamza, Yue Zhang, and Daniel Gildea. Leveraging context information for natural question generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pp. 569–574, 2018.

Ke Tran, Arianna Bisazza, and Christof Monz. The importance of being recurrent for modeling hierarchical structure. *arXiv preprint arXiv:1803.03585*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3901–3910, 2018.

A TRAINING DETAILS

We implement[1] all our models in PyTorch[2]. In HierSeq2Seq we used 2-layer BiLSTM sentence and paragraph encoders and a single-layer LSTM decoder. We set RNN hidden size to 600. For TransSeq2Seq and HierTransSeq2Seq we used 4-layer Transformer encoder and decoder with model dimension set to 256 and inner hidden dimension set to 2048.

For both the above settings we use a vocabulary of size 45k, shared across source and target of train and dev sets. We prune out all words with frequency lower than 5 in the entire dataset (train and dev). We use pretrained GloVe embeddings of 300 dimensions to represent the words in vocabulary. We use beam search decoder with beam size 5 to decode questions.

To train HierSeq2seq model, we use SGD with momentum for optimization. We set learning rate to 0.1 at the beginning, with the value being halved at every even epochs starting from epoch 8. We train our models for 20 epochs and select the model with lowest perplexity as the final model. we train HierTransSeq2seq using Adam optimizer, we set initial learning rate to 1e-7 and warmup steps to 2000 . We use standard inverse square root scheduler for scheduling the learning rate.

B EXAMPLE QUESTIONS GENERATED BY OUR BEST MODEL (HIERSEQ2SEQ + AE) ON MS MARCO TEST SET

Paragraph - lop stands for lack of prosecution . generally dismissal for lack of prosecution occurs when the court closes a case as nothing has been filed within a specified period of time . the court presumes that as there is no record of activity the party does not wish to pursue the case .

Answer - lack of prosecution .

Generated Question - what does lop mean ?

Paragraph - latin meaning : the name claire is a latin baby name . in latin the meaning of the name claire is : from the feminine form of the latin adjective ' clarus ' meaning bright or clear . also distinguished . famous bearer : twelfth century st clare (or clara) of assisi founded the poor clares order of nuns .

Answer - bright or clear .

Generated Question - what does the name claire mean ?

Paragraph - golden pheasants and silver pheasants (also a native of china) have been used symbolically in ancient chinese culture and tradition for hundreds of centuries . to the chinese the golden pheasant is a symbol of beauty , good fortune and refinement . exotic golden pheasant symbolic in chinese culture and tradition , photo melbourne zoo . in the song dynasty (960 - 1279) women wore robes decorated with colorful pheasants for important state occasions .

Answer - beauty , good fortune and refinement .

Generated Question - what does golden pheasant symbolize ?

Paragraph - distribution of the tiger . ever wonder where tigers live ! well ... most tigers live in asia , specifically throughout southeast asia , china , korea and russia . tigers like to live in swamps , grasslands , and rain forests . usually where tigers live there are trees , bushes , and clumps of tall grass

Answer - tigers live in asia , specifically throughout southeast asia , china , korea and russia .

Generated Question - where do tigers live ?

[1] Code and dataset will be released upon publication

[2] https://pytorch.org/

C SOME EXAMPLES WHERE OUR BEST MODEL (HIERSEQ2SEQ + AE) FAILS

Paragraph - 1 boil until tender [usually , 30 - 45 minutes , depending on the size of the **beets**]. 2 boiling can take up to 60 minutes for larger beets . 3 drain and run cold water over beets . 4 the skins will slip right off with the root ends , but make sure you wear kitchen gloves so your hands don't turn red .

Answer - 30 - 45 minutes

Expected Question - how long does it take to boil **beets**

Generated Question - how long does it take to boil **spaghetti**

Explanation - Model got confused because of highly frequent named entity (spaghetti) in the dataset. This mistake cannot be attributed to our model/architecture.

Paragraph - ok , the above answer is not true . as an rn in **florida** i can say that rn 's starting salary is between $ 21-$24 per hour with an average of 1 dollar pay increases per hour per year , not to mention this is before taxes are taken out . rn 's are in demand in florida , but they are not as well compensated as is rumored . on average a rn makes from $ 38,000 to $ 70,000 per year . new grads in some areas such as ny can ma ke $ 50,000 per year .... after you get a year or two under your belt in med surge , try traveling nursing ..... you can make upwards of $ 80 k to $ 90 k per year ... this assignments usually are 4 weeks to 12 weeks .

Answer - $ 38,000 to $ 70,000 per year .

Expected Question - how much does an rn make in **florida**

Generated Question - how much does a rn make in **california**

Explanation - Model got confused because of highly frequent named entity (california) in the training set. This may be because we use pre-trained GloVe embeddings. This mistake cannot be attributed to our model/architecture.
