QUESTION GENERATION USING SEQUENCE-TO-SEQUENCE MODEL WITH SEMANTIC ROLE LABELS

ALIREZA NAEIJI

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTERS OF APPLIED SCIENCE

GRADUATE PROGRAM IN ELECTRICAL ENGINEERING & COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

DECEMBER 2022

Abstract

Automatic generation of questions from text has gained increasing attention due to its useful applications. We propose a novel question generation method that combines the benefits of rule-based and neural sequence-to-sequence (Seq2Seq) models. The proposed method can automatically generate multiple questions from an input sentence covering different views of the sentence as in rule-based methods, while more complicated "rules" can be learned via the Seq2Seq model. The method utilizes semantic role labeling (SRL) used in rule-based methods to convert training examples into their semantic representations, and then trains a sequence-to-sequence model over the semantic representations. Our extensive experiments on three real-world data sets show that the proposed method significantly improves the state-of-the-art neural question generation approaches in terms of both *automatic* and *human* evaluation measures. Moreover, we extend our proposed approach to a paragraph-level SRL-based method and evaluate it on two data sets. Through both *automatic* and *human* evaluations, we show that our proposed framework remarkably improves its Seq2Seq counterparts.

Acknowledgements

I would like to express my deepest gratitude to my advisor, Dr. Aijun An. She continually provided support, guidance and invaluable feedback during my Masters at York University. Words cannot express my gratitude for her caring and passionate persona. I would also like to thank the members of my committee, Dr. Enamul Hoque Prince and Dr. Manos Papagelis, for their continuous encouragement and insightful comments, as well as Dr. Heidar Davoudi, who has supported me through not only my thesis but with several projects over my years at York.

I would like to thank iNAGO Corporation for their collaboration on this research as well as for providing me with the data set and human evaluation results. I would also like to thank Muath Alzghool and Marjan Delpisheh for their research before I joined the team. The work reported in the thesis extends their initial work on using semantic role labeling and sequence-to-sequence models for question generation.

Many thanks to my friends who are here with me and those who are far away. You all helped to make my two years even more enjoyable and memorable. Lastly, I'd like to thank my family for their unconditional support throughout my life. I would not be where I am today without their love.

This thesis work is partially funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) via an NSERC Alliance Grant and the Ontario Center of

Innovation (OCI) via a TalentEdge Internship Grant which provided support during my internship at iNAGO.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Question Generation (QG) from text has gained increasing interest due to its usefulness in various applications such as conversational systems (Ren et al., 2018), educational question generation (G. Chen et al., 2018), and reading comprehension assessment (Kumar et al., 2018; Yao et al., 2018), data augmentation for training question-answering systems (Sultan et al., 2020), and response generation in conversational systems (Gu et al., 2021). The goal of Automatic Question Generation (AQG) is to generate meaningful and natural questions from text with minimum human intervention.

1.1 Motivations

We have been working on QG for the purpose of automatically creating a Knowledge Base (KB) for the conversational QA systems of an industry partner. The KB contains a set of QA pairs extracted from a domain-specific document (such as car manuals). During question-answering, a user's question is matched against the QAs in the KB to find answers. Previously, the creation of their KB was done manually by the partner, which is very labor-intensive. To automate this KB generation process, we have tried both rule-based and neural

sequence-to-sequence (Seq2Seq) QG methods.

Traditional QG methods mainly construct heuristic rules to convert an input sentence to a question (Chali and Hasan, 2015; Flor and Riordan, 2018). The rule-based methods create rules based on linguistic features that capture the relationships among components of a sentence and can generate multiple questions from an input sentence to cover different aspects of the sentence. However, designing such rules is a very *labour-intensive* task. Also, these rules may not capture the *complexity* of ways a human asks questions (Yuan, T. Wang, A. P. Trischler, et al., 2019). As we will show in Section 5.4, the rule-based method does not lead to good results compared to neural Seq2Seq models.

Recently, various neural Seq2Seq (Sutskever et al., 2014) models have been used to generate questions from text (Du et al., 2017; Yuan, T. Wang, Gulcehre, et al., 2017; Zhao et al., 2018; Zhou et al., 2018; Bao et al., 2020; Xiao et al., 2020). These models, trained on question-answer (QA) pairs, learn to map an unseen sentence to a question. For example, in (Du et al., 2017), a seq-to-seq model with an attention mechanism was employed to generate questions. In (Zhao et al., 2018), the maxout pointer and gated self-attention network is proposed to produce questions for both sentence and paragraph level inputs.

While the Seq2Seq methods achieved better results than rule-based methods (also found in (Du et al., 2017; Zhou et al., 2018)), such methods are highly data-driven. For domains with limited training data (such as car manuals), relations that map the input text to questions cannot be well captured. In addition, Seq2Seq models often generate a single question from an input text. However, multiple questions can be asked about a piece of text from different aspects. For example, "Obama was born in Hawaii in 1961" can answer two questions: "Where was Obama born?" and "In what year was Obama born?".

One way to generate multiple questions with Seq2Seq models from an input text is to mark the input text with different short *answer spans* or keywords to show the focus for QG

so that multiple questions may be generated from the same text in multiple runs, one for each answer span/keyword. However, in our problem setting/application, such answer spans or keywords are not available as marking answer spans or keywords when creating training data requires intensive labor work. Our partner prefers an answer-unaware QG system that can automatically generate multiple factual questions without indicating answer spans or keywords in either training or inference time.

Alternatively, we may generate multiple questions given a single input sequence by using `[SEP]` tokens to separate the ground-truth questions in the output part of each training example for Seq2Seq models. In this way, the model is trained to generate multiple questions given a single input sequence. However, as we will show in Section 5, such methods significantly decrease the quality of generated questions compared to transformers that generate a single question.

Yet another technique for generating multiple questions from a source sentence is to use diverse beam search (Vijayakumar et al., 2018; J.-Y. Lin et al., 2021; Z. Zhang and Zhu, 2021). However, the beam search methods require the user to specify the number of questions returned, which is hard to specify and the ideal number of generated questions varies among different input texts.

1.2 Contributions

To address these issues, we propose a novel approach to question generation, which uses SRL, which is commonly used in rule-based systems that can label an input sentence in different ways corresponding to multiple semantic views of an input sentence, and then trains a Seq2Seq model with SRL-labeled sequences for question generation. Briefly speaking, SRL is a process that assigns labels to words or phrases in a sentence to indicate their semantic role

in the sentence with respect to a predicate, such as that of an agent, patient, goal, or result. Our method does not need keyword/answer span labels in the training data nor specifications of the number of multiple answers to be generated. The use of SRL also increases the number of training examples in the data set, which may help alleviate the problem of the limited labeled data problem.

We evaluate the proposed sentence-level method on three real-world data sets and compared the proposed method with several state-of-the-art (SOTA) QG methods including the ones that generate multiple questions. The extensive experiments on these data sets show that the proposed framework is significantly better than the SOTA Seq2Seq models and rule-based methods, especially in terms of coverage and overall scores considering both precision and recall.

Just as importantly, we extend our approach to paragraph-level QG via sentence segmentation. We evaluate the proposed paragraph-level SRL-based method on two data sets. Through both automatic and human evaluations, we show that our proposed framework remarkably improves its Seq2Seq counterparts.

1.3 Overview of the thesis

The remainder of the thesis is organized as follows: We give an overview of the development of language models and SRL in Chapter 2. We review existing works related to our research in Chapter 3. This is followed with a description of our proposed methods in Chapter 4. Evaluation of proposed approaches and comparison with SOTA methods are presented in Chapter 5. In the end, Chapter 6 concludes and summarizes our work and discusses limitations and future directions.

Chapter 2

Background

2.1 Recurrent Neural Network

In sequential tasks such as Natural Language Processing (NLP), there are always dependencies between the current input and the previous inputs. Traditional language models (e.g., N-grams) use conditional probabilities to predict the most likely sequence of words. In order to obtain the dependencies between distant words in a sequence, large N-grams are needed. Computing the probabilities of large N-grams is only limited to N previous words and requires a lot of space and memory. Yet N-gram models have a limited ability to capture longer-distance contexts.

To tackle these issues, Recurrent Neural Network (RNN) was introduced as an alternative to transmit information within a word sequence. In other words, RNN has a loop-wise architecture that passes the information from the beginning of a sequence through to the end. Besides, the computations share most of the parameters in RNN.

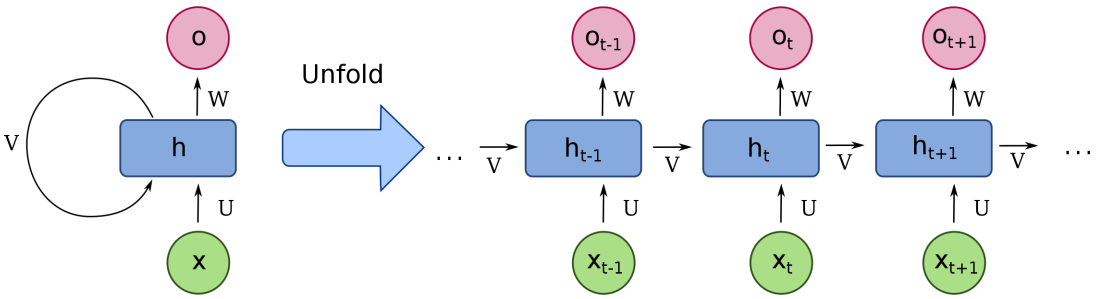


Figure 2.1: Unfolded recurrent neural network.

Figure 2.1[1] depicts the building blocks that constitute an RNN at different time steps. To get the output at time step t, the network is unfolded for $t-1$ time steps. The notations used in the figure are as follows:

- $x_t \in \mathbb{R}$ is the input at time step t. x_t is usually words or one-hot vectors being transformed into word embeddings from pre-trained vector embedding models such as Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), fastText (Joulin et al., 2017), and ELMo (Peters et al., 2018).
- $o_t \in \mathbb{R}$ is the prediction of network at time step t.
- $h_t \in \mathbb{R}^m$ holds the hidden states at time step t where m is the number of hidden states.
- $U \in \mathbb{R}^m$ are weight parameters connected with inputs in the recurrent layer.
- $V \in \mathbb{R}^{m \times m}$ are weight parameters connected with hidden states in the recurrent layer.
- $W \in \mathbb{R}^m$ are weight parameters connected with hidden units to output units.

[1] https://en.wikipedia.org/wiki/Recurrent_neural_network

The hidden state at time step t is computed with an activation function g (e.g., Sigmoid, Tanh, Relu) as follows:

$$h_t = g(x_{t-1}, h_{t-1}, U, V, b_h) = g(Ux_t + Vh_{t-1} + b_h)$$

where b_h is the bias term associated with the recurrent layer. After computing the hidden state at time step t, it is possible to get a prediction o_t with the bias term b_y associated with the feedforward layer as follows:

$$o_t = g(h_t, W, b_y) = g(W \cdot h_t + b_y)$$

The cost function used in an RNN is usually the cross entropy loss. To get the average cost in each time step, the loss over several time steps is computed using the following formula:

$$J = -\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{K} y_j^{<t>} logo_j^{<t>}$$

where K is the number of words/classes, $y_j^{<t>}$ is target word, and $logo_j^{<t>}$ is the output of softmax function. Note that the softmax function produces the probabilities for words.

2.2 Long Short Term Memory

While RNNs allow us to capture dependencies within a word sequence, they fail to exploit longer contexts (Bengio et al., 1994). In addition, RNNs are prone to vanishing or exploding gradients during the back-propagating through time (Pascanu et al., 2013). Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) is a variant of RNN designed to handle the entire sequences of data. To put it simply, LSTM receives the information from the previous timestamp and learns when to remember relevant data and when to forget useless information.

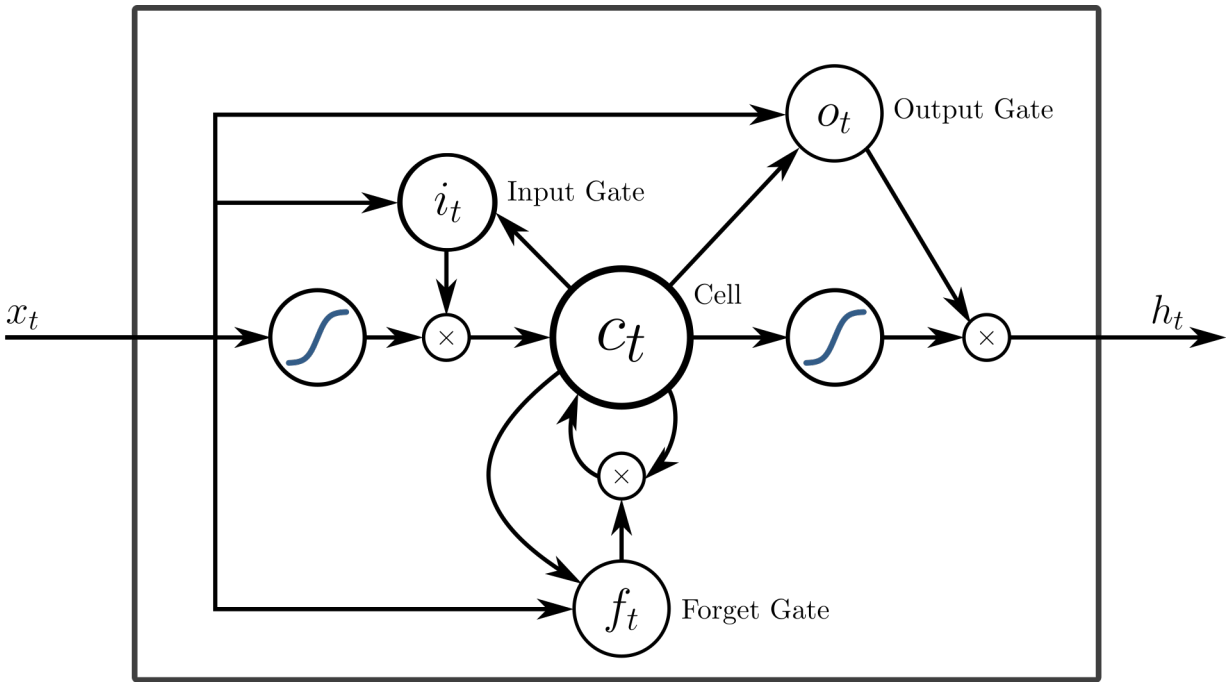


Figure 2.2: A LSTM unit with input (i), output (o), and forget (f) gates.

Figure 2.2[2] shows the structure of a typical LSTM consisting of a cell state and a hidden state with three gates namely input (i), output (o), and forget (f) gates. The hidden state holds the output from the LSTM cell at time step t and can be formally written using the activation function k:

$$h_t = k(h_{t-1}, x_t)$$

where h_{t-1} is the hidden state from previous time step and x_t is word input representation at current time step t. The computations behind it are described below.

- Forget gate chooses which information from the previous cell state and current input should be kept. Using the Sigmoid function results in an output value between zero and one. A value close to zero indicates to discard the information while close to one

[2]https://en.wikipedia.org/wiki/Long_short-term_memory

means to preserve it:

$$f_t = \sigma(W_{fh}h_{t-1} + W_{fx}x_t + b_f)$$

- Input gate decides which values to update by yielding a value between zero and one that value closer to one has higher importance to be preserved. g_t is a regulation layer to help the flow of information by squeezing values within -1 to 1:

$$i_t = \sigma(W_{ih}h_{t-1} + W_{ix}x_t + b_i)$$

$$g_t = tanh(W_{gh}h_{t-1} + W_{gx}x_t + b_g)$$

- Output gate determines how much information to pass over to the output:

$$o_t = \sigma(W_{oh}h_{t-1} + W_{ox}x_t + b_o)$$

- Cell state carries all the relevant information through the sequence:

$$c_t = f_t \cdot c_{t-1} + i_t \cdot g_t$$

Note that all weights (e.g., W_{fh}, W_{fx}, W_{ih}, W_{ix}, W_{gh}, W_{gx}, W_{oh}, W_{ox}) and biases (e.g., b_f, b_i, b_g, b_o) are learnable parameters during the training phase.

2.3 Transformer Models

LSTMs are used to overcome the vanishing and exploding gradient problem but they require the sequential data to be processed in order. The more words there are in the input sequence, the longer time it takes to process the input. On the other hand, processing long sentence sequentially, even with LSTMs, causes the information to get lost within the network. Transformer (Vaswani et al., 2017) is a Seq2Seq model based on a self-attention mechanism and specially designed to avoid this entire recurrence for machine translation task.

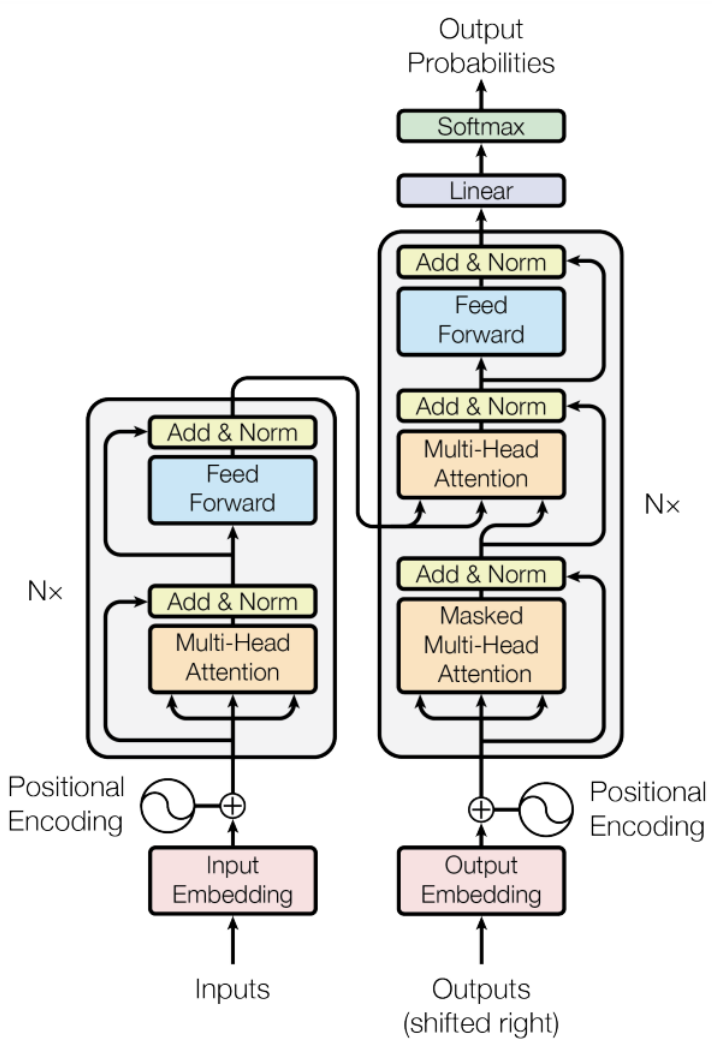


Figure 2.3: Transformer model architecture (Vaswani et al., 2017).

Figure 2.3 illustrates the architecture of Transformer. As a black box, it looks like an encoder and a decoder which the encoder first encodes the sentence from the source language into a representation, and then the decoder decodes it into the target language. The encoder and decoder have very similar components except for the first layer of the decoder.

2.3.1 Transformer Encoder

The encoder component is made up of N = 6 identical layers. The encoder can become separated into 2 parts: (1) multi-head self-attention module, and (2) position-wise fully connected feed-forward network. A residual connection (He et al., 2016) is applied around each of the two parts, followed by layer normalization (Ba et al., 2016). Self-attention mechanism enables the model to focus on the relevant parts of the input sequence. Given a word sequence X of length n represented by word embeddings $[x_1, x_2, .., x_n]$, there are 4 steps inside self-attention module:

- Query, key, and value vectors are created for these words by multiplying vector x_i with weight matrices which are trained during training. This step can be done in parallel for all tokens.

$$[q_1, q_2, .., q_n] = [x_1, x_2, .., x_n] \times W^q$$

$$[k_1, k_2, .., k_n] = [x_1, x_2, .., x_n] \times W^k$$

$$[v_1, v_2, .., v_n] = [x_1, x_2, .., x_n] \times W^v$$

- The dot product between query and key vectors of token x_i and each word is calculated. This step gives us the similarity score between words and decides which word, x_i is going to pay more attention to.

$$[s_1, s_2, .., s_n] = q_i \cdot [k_1, k_2, .., k_n]$$

- The similarity vector is divided by the square root of the dimension of the key vectors. It is because to have more stable gradients. Then, the result is passed through a softmax operation to produce probability distributions. It could be expressed that the token x_i associates with itself with the probability of a_i.

$$[a_1, a_2, .., a_n] = softmax(\frac{[s_1, s_2, .., s_n]}{\sqrt{d_k}})$$

- The softmax score vector from the previous step is multiplied by the value vector of token x_i. Next, the weighted value vectors are summed up and the output vector is produced.

$$z_i = \sum_{1}^{n}([a_1, a_2, .., a_n] \times v_i)$$

The calculation above can be done in matrix form for quicker processing:

$$Z = softmax(\frac{Q \times K^T}{\sqrt{d_k}}) \times V$$

where Q, K, and V are query, key and value vectors respectively. The matrix Z is the self-attention score which tells us how much a word interacts with other words. This is the intuition behind multi-head self-attention that each head behaves with the same word differently. The heads are just multiple copies of identical self-attention modules.

RNNs know about the position of each word in a sequence because it is a sequential process. However, the transformer does not process sentences sequentially (it happens at once) and each word is independent of the other ones. To fix this issue, *positional encoding* was proposed. For words to represent different positions, positional encoding is added to their word embedding. These positional encodings are either learned during training or fixed that comes from sine and cosine functions.

2.3.2 Transformer Decoder

Once we have the contextual representations of our source sentence, we can feed them into the decoder. The decoder component is also made up of N = 6 identical layers. As mentioned, the encoder and decoder have very similar components except for the first layer of the decoder. In the first layer, there is a masked multi-head self-attention instead of regular multi-head self-attention due to the fact that the self-attention in the decoder can only attend to the previous words.

2.4 Semantic Role Labeling

Semantic Role Labeling (SRL), a.k.a shallow semantic parsing, recognizes the predicate-argument structure of a sentence and assigns labels (i.e., semantic roles, such as Agent, Patient, Instrument, Beneficiary, Goal or Result) to words or phrases in a sentence. The predicate-argument structure basically explains the meaning of a sentence in the form of who did what to whom, when, where, and how. This process involves identifying the semantic arguments related to the predicate (usually a verb) of a sentence and classifying them into their specific roles. In a nutshell, the task of SRL is to discover the predicate and specify how these arguments are semantically related to the predicate.

For example given the sentence *"Alex bought a book from Sara last month"*, the predicate is *'bought'* and its arguments are as follows:

- *'Alex'* is [ARG0] representing the agent (the person doing something in the event).
- *'a book'* is [ARG1] representing the patient (something underwent a change of state during the course of the event).
- *'from Sara'* is [ARG2] representing benefactive (the person that benefits from the event).
- *'last month'* is [ARGM-TMP] representing time (temporal modifier of the event).

The semantic analysis of a sentence is an important step for downstream NLP tasks in understanding the meaning of it. SRL is used in many NLP tasks involving text understanding such as information retrieval systems (S. Chen et al., 2020), text summarization (Khan et al., 2015), reading comprehension (Liu et al., 2021), and machine translation (Song, Gildea, et al., 2019).

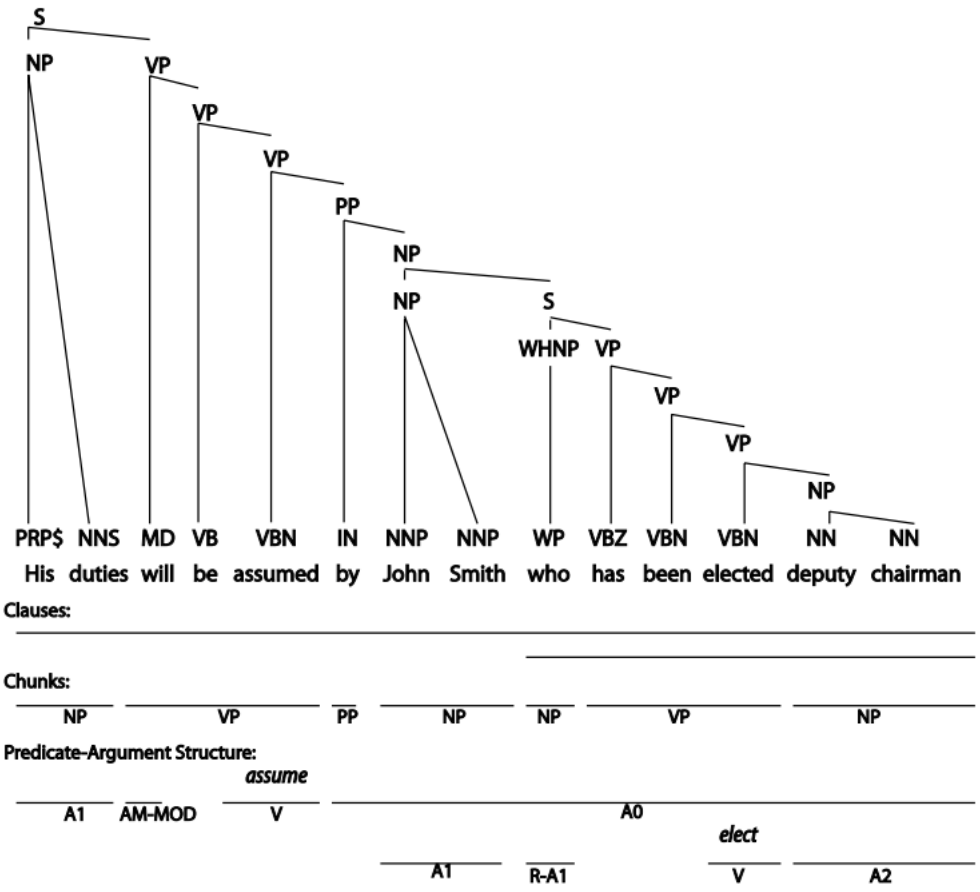


Figure 2.4: An example of a parse tree and its predicate–argument structure (Punyakanok et al., 2008)

.

Traditional semantic role labelers are feature-based (e.g., predicate and POS tag of predicate, voice, position, path) where syntax appears to be a prerequisite for SRL extraction (Punyakanok et al., 2008). They heavily rely on shallow syntactic parsers (i.e., chunkers and clausers) or full syntactic parsers (using full parse trees). Both parsers try to find syntactic relations between word pairs. Figure 2.4 provides an example of chunks, clauses (obtained by a parse tree) and its predicate–argument structure. Another example is *Clear Parser SRL* (Choi and Palmer, 2011) where the semantic labels are generated for the input

text based on a transition-based SRL approach.

Recently end-to-end deep learning models either based on LSTM (Li et al., 2019) or Transformer (Papay et al., 2022) accomplished SOTA performance without leveraging any syntactic dependencies. An example includes *SRL BERT* (Shi and J. Lin, 2019) provided in AllenNLP (Gardner et al., 2017) to produce the semantic labels for the input text, which is a SOTA model for SRL extraction. This method generates a predicate-argument structure for a sentence based on a BERT-based approach. In this model, each sentence is represented by one or more propositions, consisting of a predicate and its semantic arguments. For example, the semantic representation of sentence *"ABS is activated during braking under certain road or stopping conditions"* is "[ARG1] *is activated* [ARGM-TMP]", where [ARG1] (representing patient) and [ARGM-TMP] (representing time) are semantic role labels for *ABS* and *during braking under certain road or stopping conditions*, respectively. Table 4.1 provides more examples with their semantic representations.

Chapter 3

Related Work

The question generation approaches broadly fall into two categories: rule-based approaches and neural Seq2Seq learning approaches.

3.1 Rule-based Question Generation System

Rule-based approaches mainly rely on hand-crafted templates/rules built upon linguistic features (Chali and Hasan, 2015; Flor and Riordan, 2018; Khullar et al., 2018; Lindberg et al., 2013). These methods use rigid heuristic rules to transform a source sentence into one or more questions. For example, in (Chali and Hasan, 2015) a set of general-purpose rules based on named entity information and the predicate-argument structures of the sentences (along with semantic roles) are used to generate questions.

In (Flor and Riordan, 2018) both wh-questions and yes/no questions are directly generated from semantic analysis of source sentences using SRL (without templates). The semantic role labels are produced from SENNA system (Collobert et al., 2011) after performing standard NLP pipeline (e.g., tokenization, POS tagging) on the samples in the dataset. The SRL output goes through a postprocessing step to correct several system-generated issues (e.g.,

wrong labels assigned by SENNA). In order to generate wh-questions, the focal argument centered around the predicate of the sentence is chosen as the answer. It means a question would be created to ask about the text of the focal argument. This step is followed by identifying the appropriate question word. Next, the question is formulated by reorganizing the remaining arguments of the predicate.

A benefit of rule-based methods is that rules are transparent and relatively easier to understand compared to neural-based methods. Also, rule-based methods allow multiple sentence views, that is, it is possible to generate multiple questions from different aspects of a sentence.

3.2 Neural Seq2Seq Question Generation

Rules have limited power in expressing the complicated mapping function that human uses for question generation. Designing a comprehensive set of rules is a very labour-intensive task. Moreover, these approaches highly depend on the lexical analyzer, syntactic parsers, or semantic role labelers and often produce meaningless or vague questions due to errors propagated from these parsers.

Introduced for the first time by Google (Sutskever et al., 2014), Neural Seq2Seq models, comprised of an encoder and a decoder, have been successfully applied to question generation due to their capability to extract effective features and model complicated functions. The encoder takes sequences of data (e.g., words) as input and produces an intermediate state in a vector space representing the semantics of the input sequence. Then, the decoder takes the state as the input and generates the output sequence accordingly. Different neural network architectures such as LSTM (Hochreiter and Schmidhuber, 1997), GRU (Cho et al., 2014), and Transformer (Vaswani et al., 2017) have been used as decoders and encoders.

Early Seq2Seq-based QG models are based on RNN structures. Examples include an LSTM-based Seq2Seq model with the global attention mechanism (Du et al., 2017) and LSTM-based model with the maxout pointer and gated self-attention network (MP-GSN) (Zhao et al., 2018). The encoder in MP-GSN is a two-layer bidirectional LSTM. A gated self-attention mechanism (W. Wang et al., 2017) is applied to the encoder to capture dependencies within the passage hidden states and thus improve the encoded representation. The decoder is a two-layer unidirectional LSTM. To avoid repetitions in the output sequence, especially when the input sequence is long, a maxout pointer mechanism is used to calculate the score of copying word from the input sequence at each step. The aforementioned model is able to handle rare words not occurring in the training data and outperforms previous approaches on QG such as (Du et al., 2017) and works better than the *vanilla* Seq2Seq model (Kumar et al., 2018).

More recent Seq2Seq models are based on Transformer (Explained in Section 2.3) which relies entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs. Text-to-Text Transfer Transformer (T5) (Raffel et al., 2020) is a multi-task pre-trained Transformer-based model, which treats a diverse set of NLP tasks as a *text-to-text* problem, and can be fine-tuned over various tasks such as classification, regression, text summarization, machine translation, and question answering. In order for T5 to perform a specific task, the model is given an input string of texts that contains both the task and the data. Note that there is one T5 model that performs various tasks without changing anything but the input sentences. BART (Lewis et al., 2020) is another cutting-edge pre-trained Transformer-based model that has a denoising autoencoder architecture. BART is specially designed for text generation tasks and trained by (1) corrupting text with an arbitrary noising function, and (2) learning a model to reconstruct the original text. Further, ProphetNet (Qi et al., 2020) is a Transformer-based Seq2Seq model that is pre-trained over

very large datasets to conduct future n-gram prediction with an n-stream self-attention mechanism. The future n-gram prediction strategy enables the system to foresee the future tokens and it also helps the model predict multiple future tokens. It is applied to QG and achieves SOTA results. Some other examples of Transformer-based QG models include PEGASUS (J. Zhang et al., 2022), UNILM (Dong et al., 2019), UNILMv2 (Bao et al., 2020), and Ernie-Gen (Xiao et al., 2020).

3.3 Diverse Question Generation

While the Seq2Seq methods achieve better results than rule-based methods (Du et al., 2017), they are highly data-driven. For domains where labelled training data are limited, relations that map the input text to questions cannot be successfully acquired. In addition, Seq2Seq models often generate a single question given an input text, which does not cover multiple views of a sentence.

To solve this single-question-generation problem, different strategies have been proposed. One strategy is to use diverse beam search (Vijayakumar et al., 2018; Z. Zhang and Zhu, 2021) and sampling techniques (such as top-p nucleus sampling used in (Sultan et al., 2020)). While these methods showed promising results, the user has to specify the bin/sample size and the number of questions to be generated (whose ideal number may depend on the input sequence).

Another strategy for generating diverse questions is to mark or extract keywords in the input text and generate questions by conditioning on keywords or keyword positions (e.g., (Pan et al., 2020; Shen et al., 2020; Subramanian et al., 2018; Sun et al., 2018; Song, Z. Wang, et al., 2018; Z. Zhang and Zhu, 2021)). However, extracting keywords (either automatically or manually) to build keyword-labeled training data often needs domain knowledge, a pre-defined

keyword list, or documents beyond the training data. The QG method we propose solves the above problems by learning Seq2Seq models with semantic role labeled QAs. It can generate multiple and diverse questions without specifying the number of questions to be generated or requiring keyword-labeled training data.

A notable method that uses both diverse beam search and keyword-based method is KPCNet (Z. Zhang and Zhu, 2021) for generating clarification questions from product descriptions in e-commerce websites. It first extracts the keywords from the title and description of a product and then uses classical or diverse beam search to generate questions conditioned on the keywords.

A recent method that also uses SRL and Seq2Seq models is a 2-step method in (Pyatkin et al., 2021). It tackles role question generation that, given a predicate mention and a passage, generates a set of questions asking about all possible semantic roles of the predicate. It first generates prototype questions for all the roles based on the ontology in PropBank (Palmer et al., 2005). It then trains a BART model to generate all questions (including ones that cannot be answered by the input text) given these prototype questions contextualized over the input text. Both the problem definition and the methodology are very different from ours. Regarding the problem definition, it generates questions targeting all semantic roles of a predicate specified in PropBank, which may not be answered by the input text. Our method does not need to generate prototype questions and we generate only the information-seeking questions that can be answered by the input text. Regarding the methodology, it lacks semantic mappers used for the transformation of SRL labels into words/phrases or vice versa. Such semantic mappers, as we will show in Sections 4.3 and 4.5, are the main part of our method and also help us satisfy the constraint of generating questions where the answer must be in the input sequence.

Chapter 4

Methodology

Given a set of answer (i.e., sentence) and question pairs, our goal is to train a model to generate from an unseen sentence one or more questions that can be answered by the sentence. As we will describe in Section 4.6, our approach can be extended to paragraph-level SRL-based method to incorporate answer with more than one sentence (i.e., paragraph, itemized, table samples).

4.1 Overview of the Method

Our method contains a *Semantic Role Labeler (SRLer)*, a Seq2Seq model, and two semantic mappers (namely, *Question2SRL* and *SRL2Question*). First, *SRLer* extracts semantic representations (i.e., SRL labels) from answers in the training set. Then *Question2SRL* maps questions in the training set to their corresponding semantic representations. Next, a Seq2Seq model is trained using these semantic representations to convert an SRL representation of an answer to that of a question. In the inference stage, *Semantic Role Labeler* extracts semantic representations ($\widehat{a}_{sem}$) of an answer $\widehat{a}$. Then, $\widehat{a}_{sem}$ is converted to an SRL representation of a question ($\widehat{q}_{sem}$) by the learned Seq2Seq model. Finally, *SRL2Question* converts $\widehat{q}_{sem}$ into a

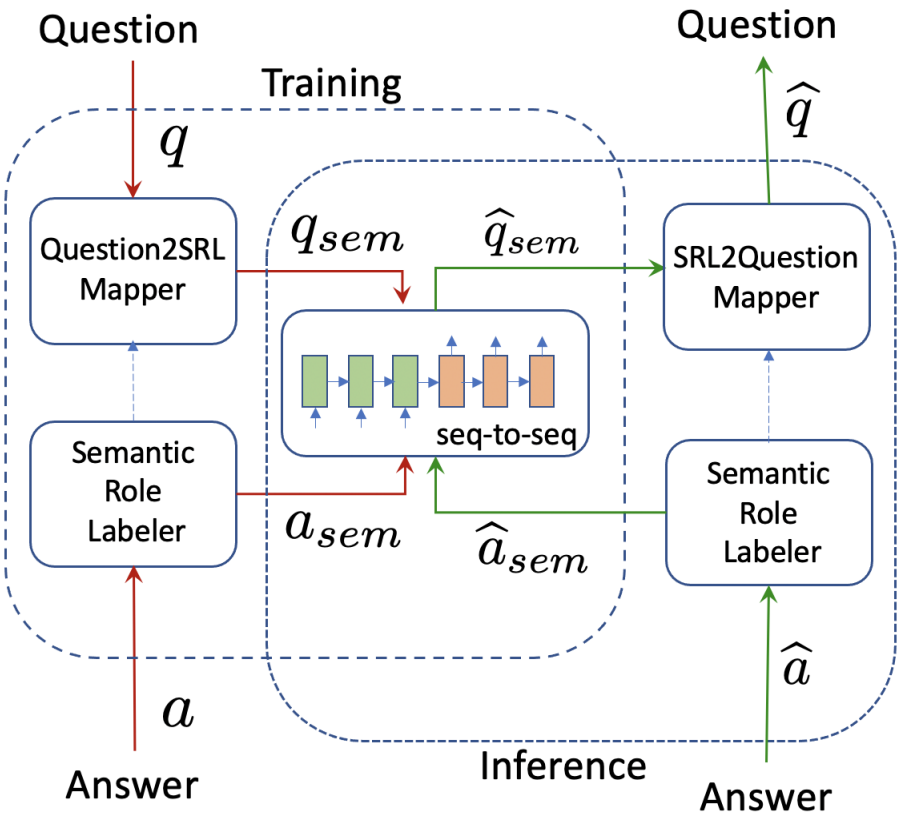


Figure 4.1: Overview of Proposed Framework

natural language question $\widehat{q}$. Figure 4.1 illustrates our proposed framework.

Next, a Seq2Seq model is trained using these semantic representations to convert an SRL representation of an answer to that of a question. In the inference stage, *Semantic Role Labeler* extracts semantic representations ($\widehat{a}_{sem}$) of an answer $\widehat{a}$. Then, $\widehat{a}_{sem}$ is converted to an SRL representation of a question ($\widehat{q}_{sem}$) by the learned Seq2Seq model. Finally, *SRL2Question* converts $\widehat{q}_{sem}$ into a natural language question $\widehat{q}$.

4.2 Semantic Role Labeler

A Semantic Role Labeler (SRLer) is used in both training and inference. In the training phase, an SRLer is used to convert each answer a in the training set into its semantic representation

(denoted as a_{sem}) that contains semantic role lables (SRL).

An SRLer recognizes the predicate-argument structure of a sentence and assigns labels (i.e., semantic roles, such as agent, goal or result) to words or phrases in a sentence. We use both *SRL BERT* (Shi and J. Lin, 2019) and *Clear Parser SRL* (Choi and Palmer, 2011) in our experiments. Since *SRL BERT* leads to better outcome, we report the results from *SRL BERT* in this thesis.

Table 4.1: Semantic representation of answers and questions (ARG0: agent , ARG1: patient, ARG2: attribute, ARGM-NEG: negation, ARGM-PRP: purpose)

Semantic representation for sample input sentences (answers): S1. [ARG1: the fuel filler funnel] is [ARG2: under the luggage compartment floor covering] . S2. [ARG0: this vehicle] has a capless refueling system and does [ARGM-NEG: not] have [ARG1: a fuel cap] . S3. distribute [ARG1: the trailer load] [ARGM-PRP: so 10 - 15 % of the total trailer weight is on the tongue] . S4. distribute the trailer load so [ARG1: 10 - 15 % of the total trailer weight] is [ARG2: on the tongue] .
Semantic representation for the questions corresponding to the above sentence representations in the training data: Q1: where is [ARG1: the fuel filler funnel] ? Q2: does [ARG0: this vehicle] have [ARG1: a fuel cap] ? Q3: how much of [ARG1: the trailer load] should be on the tongue ? Q4: how much of the trailer load should be [ARG2: on the tongue] ?

Table 4.1 provides more examples of semantic representations for answers. Note that if a sentence contains more than one verb, more than one semantic representation may be generated. For example, S3 and S4 in Table 4.1 are from the same sentence.

4.3 The Question2SRL Mapper

The *Question2SRL* mapper converts a question q in the training data into its semantic representation q_{sem}. Instead of applying an SRLer directly on q, *Question2SRL* uses the semantic role labels in the semantic representation a_{sem} of q's corresponding answer a to label the phrases or words in q. We design two approaches for *Question2SRL*, namely, *Hard-Question2SRL* and *Soft-Question2SRL*:

Hard-Question2SRL*:* In this approach, for each semantic role label l that occurs in an answer's SRL representation, if its corresponding phrase or word occurs in the question, the phrase or word in the question is replaced with label l. The reason why we did not use semantic role labeling to directly label the question is that we would like to keep the question words (e.g., what, where, when, etc.) in the semantic representation of the question, and also that semantic role labeling may generate labels for a question which do not occur in its answer. The lower part of Table 4.1 provides the semantic representations of the questions corresponding to the answer SRL representations in the upper part of the table. Note that Q3 and Q4 are two different representations of the same question, resulting from two different SRL representations of the same answer (S3 and S4). Thus, one original training example can be converted to one or more SRL-labeled examples for training a Seq2Seq model, resulting in an increase in the size of training data.

Soft-Question2SRL*:* The words/phrases labeled with a semantic role in an answer may not occur exactly in the question, but their synonyms or similar expressions may. In this case, *Hard-Question2SRL* may not find the exact match in the question. To address this issue, we design *Soft-Question2SRL* that considers the semantic similarity between the words/phrases corresponding to a semantic role label in a_{sem} and potential words/phrases in a question to find the best match. Algorithm 1 outlines the procedure. Given a set of SRLs (L) generated

for answer a by *Semantic Role Labeler*, and question q, we first generate all possible n-grams (nG) from q, then compare the phrases/words corresponding to each label $l \in L$, denoted by $words(l)$, with each n-gram $ng \in nG$. The n-gram with maximum similarity with $words(l)$ is selected to be replaced by l as long as the similarity is greater than or equal to a threshold α. We calculate the similarity between n-grams and semantic role label words based on cosine similarity between their corresponding *Sentence-BERT* embeddings (Reimers and Gurevych, 2019).

Algorithm 1: *Soft-Question2SRL*

Input : L // a set of SRLs generated for a

q // question

Output : q_{sem} // semantic rep. of q

$nG \leftarrow$ Generate all n-grams from q

for *each* $l \in L$ **do**
  $score \leftarrow 0$
  for *each* $ng \in nG$ **do**
    $sim \leftarrow \text{Cosine}(words(l), ng)$
    if $score < sim$ **then**
      $score \leftarrow sim$
      $l_{best} \leftarrow l$
  if $score \geq \alpha$ **then**
    $q_{sem} \leftarrow$ replace ng in q with l_{best}

Table 4.2 shows how the algorithm works using an example. The first box of the table illustrates the answer a, and its respective question q. Each subsequent box shows a semantic role label $l \in L$ in a_{sem}, the best n-gram ng matching with $words(l)$, and their respective similarity score. The final box shows the semantic representation of question q_{sem} after replacing n-grams with SRLs in previous steps. Note that we replace the n-gram with an

Table 4.2: An example of *Soft-Question2SRL* mapper converting a question into its semantic representation. Each row shows the best score and corresponding n-gram in the question ($\alpha = 0.85$).

Semantic representation (a_{sem}) for sample answer sentence a and its corresponding question q: a_{sem} = [ARGM-TMP: in january 2009] , [ARG0: the green power partnership -lrb- gpp , sponsored by the epa -rrb-] [V: listed] [ARG1: northwestern] [ARG2: as one of the top 10 universities in the country in purchasing energy from renewable sources] . q = in 2009 , who named northwestern as one of the top 10 universities in the country in purchasing renewable energy ?
l = [ARGM-TMP], $words(l)$ = "in january 2009" *Best matching ng* = "in 2009", *Similarity score* = $0.91 > \alpha \Rightarrow$ **Replace** ng **with** l **in** q_{sem}
l = [ARG0], $words(l)$ = "the green power partnership -lrb- gpp , sponsored by the epa -rrb- *Best matching ng* = "in purchasing renewable energy ?", *Similarity score* = $0.46 < \alpha$
l = [V], $words(l)$ = " listed" *Best matching ng* $\leftarrow$ "in", *Similarity score* = $0.42 < \alpha$
l = [ARG1], $words(l)$ = "northwestern" *Best matching ng* = "northwestern", *Similarity score* = $1.00 > \alpha \Rightarrow$ **Replace** ng **with** l **in** q_{sem}
l = [ARG2], $words(l)$ = "as one of the top 10 universities in the country in purchasing energy from renewable sources" *Best matching ng* = "as one of the top 10 universities in the country in purchasing renewable energy", *Similarity score* = $0.98 > \alpha \Rightarrow$ **Replace** ng **with** l **in** q_{sem}
q_{sem} = ARGM-TMP , who named ARG1 ARG2 ?

SRL label if the score is higher than or equal to a threshold (i.e., α).

4.4 Sequence-to-Sequence Learning

Given a set of $\langle a_{sem}, q_{sem} \rangle$ pairs (where a_{sem} and q_{sem} are an SRL-labeled answer and an SRL-labeled question, respectively), we train a Seq2Seq model to convert a_{sem} to q_{sem}. Any Seq2Seq model can be used for this purpose. In our experiment, we use the state-of-the-art models T5 (Raffel et al., 2020) and BART (Lewis et al., 2020) to evaluate our method.

We feed the Seq2Seq model with a_{sem} as the input sequence, where some of words/phrases in the original answer a are replaced by SRLs. While SRLs provide the Seq2Seq model with useful information, this replacement may reduce the information to which model is exposed. As another strategy, we add the actual answer a as a context to the input sequence. That is, the input to the Seq2Seq model is $\langle answer\text{: } a_{sem}\ context\text{: } a \rangle$, where *answer*: and *context*: are tokens prepended to a_{sem} and a, respectively. Alternatively, we can add the SRL labels of the actual answer a and their corresponding words separated by a special token <*sep*> to the input sequence. The input to the Seq2Seq model is $\langle answer\text{: } a_{sem}$ <*sep*> *label words* <*sep*>$\rangle$.

Table 4.3: A training example for Seq2Seq model with 3 different strategies for input (i.e., answer) representations.

input	*answer*: [ARG1] was named [ARG2], [ARGM-PRD].
input+C	*answer*: [ARG1] was named [ARG2], [ARGM-PRD]. *context*: denver linebacker von miller was named super bowl mvp, recording five solo tackles, 2 1/2 sacks, and two forced fumbles.
input+L	*answer*: [ARG1] was named [ARG2], [ARGM-PRD]. <*sep*> [ARG1] denver linebacker von miller <*sep*> [ARG2] super bowl mvp <*sep*> [ARGM-PRD] recording five solo tackles , 2 1/2 sacks , and two forced fumbles <*sep*>
output	who won the [ARG2] ?

Table 4.3 shows a training example with three variations of the input. We will compare the three input versions of the Seq2Seq model in the experiment.

4.5 The SRL2Question Mapper

In the inference phase, the SRLer is first used to convert an input sentence (i.e., answer $\widehat{a}$) into its semantic representation ($\widehat{a}_{sem}$). Then, the trained Seq2Seq model is used to convert $\widehat{a}_{sem}$ into a semantic representation of a question ($\widehat{q}_{sem}$). After that, the *SRL2Question* mapper transforms all the semantic role labels in the generated semantic representation $\widehat{q}_{sem}$ into words or phrases. In particular, for each semantic role label l in the semantic representation $\widehat{q}_{sem}$ generated by the Seq2Seq model, the *SRL2Question* mapper looks for label l in all the semantic representations $\widehat{a}_{sem}$ of the input sequence $\widehat{a}$, and uses the phrase or word in $\widehat{a}$ that corresponds to l to replace l in $\widehat{q}_{sem}$.

Table 4.4 shows examples of generated semantic representations and converted questions, together with their input sentences, semantic representations of the input sentences, and ground truth questions.

4.6 Extension to Paragraph-level Method

We observe that as input texts get longer and more complicated, the SRL model struggles with generating more accurate output. Hence, we extend our approach to paragraph-level QG method via sentence segmentation.

Given a set of training data consisting of question and answer pairs, we split answers into separate sentences using NLTK (Bird and Loper, 2004) and keep the corresponding ground truth question along with each sentence. NLTK is a toolkit for natural language processing systems in English. This would enlarge the training data since the answer may contain more than one sentence. Our paragraph-level dataset is reduced to a sentence-level one where we can use our proposed approach as discussed in 4.1 to tackle this matter. Before feeding the Seq2Seq model with a_{sem} as the input sequence, we add the actual paragraph p as a context

Table 4.4: Examples of sentence $\widehat{a}$, its semantic representation $\widehat{a}_{sem}$, the outcome $\widehat{q}_{sem}$ generated by Seq2Seq, the question $\widehat{q}$ converted from $\widehat{q}_{sem}$, and ground-truth question Q_t from the Car Manuals dataset.

$\widehat{a}$:	before placing a child in the child restraint , make sure it is securely held in place .
$\widehat{a}_{sem}$:	before placing [ARG1] [ARG2] , make sure it is securely held in place .
$\widehat{q}_{sem}$:	what should i do before placing [ARG1] [ARG2] ?
$\widehat{q}$:	what should i do before placing a child in the child restraint ?
Q_t:	what should i do before placing a child in the child restraint ?
$\widehat{a}$:	adjust the temperature setting using the + and - temperature buttons on the right-hand side of the climate controls .
$\widehat{a}_{sem1}$:	adjust the [ARG1] setting using the + and - temperature buttons on the right-hand side of the climate controls .
$\widehat{a}_{sem2}$:	adjust the temperature setting using [ARG1] [ARGM-LOC] .
$\widehat{q}_{sem1}$:	how do i adjust the [ARG1] setting ?
$\widehat{q}_{sem2}$:	how do i adjust the temperature [ARGM-LOC] ?
$\widehat{q}_1$:	how do i adjust the temperature setting ?
$\widehat{q}_2$:	how do i adjust the temperature on the right-hand side of the climate controls ?
Q_t:	how do i adjust the temperature on the passenger 's side ?

to the input sequence. That is, the input to the Seq2Seq model is $\langle answer\text{: } a_{sem}\ context\text{: } p\rangle$, where *answer*: and *context*: are tokens prepended to a_{sem} and p accordingly. Then, a Seq2Seq model is fine-tuned to convert an SRL representation of an input sentence along with its original paragraph into an SRL representation of a question, which is then converted to a natural language question.

Chapter 5

Empirical Evaluation

We investigate (1) whether using SRL with Seq2Seq models improves the QG performance of Seq2Seq without SRL, (2) how our method compares with the state-of-the-art QG methods, and (3) if our SRL-based QG method extended to paragraph-level results in improvements over baseline.

5.1 Datasets

We evaluate our method on 3 datasets. The first one contains QAs created by human annotators from two car manuals of Ford and GM, denoted as Car Manuals. The second dataset is SQuAD (Rajpurkar et al., 2016), containing QAs created by Amazon Mechanical Turk crowd-workers from Wikipedia articles. We use the processed sentence-level SQuAD dataset (Du et al., 2017), where the answer in a QA pair is a single sentence. The third dataset is NewsQA (A. Trischler et al., 2016), a machine comprehension dataset of human-generated QA pairs from CNN news articles. We created a sentence-level NewsQA dataset. The sentences were extracted from corresponding paragraphs based on their answer span. All the datasets contain training, testing, and development sets. Their statistics are given in Table

5.1. In these datasets, multiple QA pairs may have the same answer sentence but different questions.

Table 5.1: Statistics of Three Datasets

Dataset	training set	test set	development set
Car Manuals	9,184 QAs	1,869 QAs	1,403 QAs
SQuAD	70,484 QAs	11,877 QAs	10,570 QAs
NewsQA	91,536 QAs	5,067 QAs	5,136 QAs

5.2 Automatic Evaluation Metrics

The original design of BLEU (Papineni et al., 2002), ROUGE (C.-Y. Lin, 2004) and METEOR (Denkowski and Lavie, 2014) scores assumes different ground-truths for the same input sequence have the same meaning, and the best match between the output sequence and one of the ground truths is used to compute the scores. However, in question generation, the multiple ground truth questions for an input sentence often have different meanings and the generated questions should cover all the ground truth questions ideally.

To overcome this limitation, we use the *precision*, *recall* and F scores proposed in (Schlichtkrull and Cheng, 2020) for measuring the quality and diversity of generated questions. Given a test example consisting of input text a and a set of ground-truth questions T, the *precision* and *recall* of a set of questions G generated from a by a QG method are defined as:

$$precision(G, T, s) = \frac{1}{|G|} \sum_{g \in G} \max_{t \in T} s(g, t)$$

$$recall(G, T, s) = \frac{1}{|T|} \sum_{t \in T} \max_{g \in G} s(g, t)$$

$$F(G,T,s) = \frac{2 \times precision(G,T,s) \times recall(G,T,s)}{precision(G,T,s) + recall(G,T,s)}$$

where s is a scoring function $s : \Omega \times \Omega \rightarrow \mathbb{R}$ that measures the similarity between two questions. We use the similarity function used in BLEU-n (n-gram text overlap), ROUGE-L (longest common subsequence text overlap) and METEOR (which takes into account word re-ordering, stemming, synonyms and paraphrase matching) to compute s. *F-score* is defined as the harmonic mean of precision and recall. Note that these precision, recall and F-scores are the same as the u, v and F scores proposed in (Schlichtkrull and Cheng, 2020) for measuring the quality and diversity of generated questions.

The recall metric measures how much of the ground-truth questions are covered by the generated questions given an answer sentence. Since the ground-truth questions in our datasets cover different aspects of an input sentence, this recall value indicates the percentage of these different/diverse ground-truth questions being covered by the generated questions. It does not measure the "diversity" among the generated questions, rather it measures the coverage of the diverse ground-truth questions by the generated questions. The measure was proposed in (Schlichtkrull and Cheng, 2020), where it was called a diversity measure. Other diversity measures, such as Distinct-3 used in (Z. Zhang and Zhu, 2021), do not measure the relevance of the generated questions to the ground-truths. For example, Distinct-3 calculates the number of distinct 3-grams in the generated questions, free of any reference or ground truth sentences. We emphasize the coverage over diverse ground truths.

5.3 Comparison of Seq2Seq+SRL with Seq2Seq methods

To investigate whether the use of SRL with Seq2Seq models improves the QG performance of Seq2Seq models trained with original sentences, we conducted experiments with two

state-of-the-art transformer-based Seq2Seq models: (1) **BART** (Lewis et al., 2020) and (2) **T5** (Raffel et al., 2020).

We fine-tune the Huggingface pre-trained models (Wolf et al., 2019) of BART-base and T5-small with 139 and 60 million parameters respectively. We use the smallest available model sizes of BART and T5 to avoid GPU memory errors. Four V100-SXM2 32GB GPUs are used for fine-tuning. For baselines, T5 and BART are fine-tuned using the original sentence-based QA pairs in each training set to generate a question given an answer.

For our method, we use a pre-trained SRL BERT model (Shi and J. Lin, 2019) provided in AllenNLP (Gardner et al., 2017) as the Semantic Role Labeler to convert answer sentences to their SRL representations. We then fine-tune T5 or BART to learn a model that maps an SRL representation of an input sentence to that of the question, which is then mapped to a natural language question.

To determine the values of hyperparameters in the Seq2Seq model (T5 and BARR) in either baseline or our model, we conduct random search (Bergstra and Bengio, 2012) due to its efficiency. To do so, we randomly select 10 combinations of learning rates (LR) and epoch numbers (EP) as follows:

- "LR $= 5 \times 10^{-6}$, EP $= 2$"
- "LR $= 10^{-5}$, EP $= 8$"
- "LR $= 5 \times 10^{-5}$, EP $= 5$"
- "LR $= 5 \times 10^{-5}$, EP $= 10$"
- "LR $= 10^{-4}$, EP $= 10$"
- "LR $= 10^{-4}$, EP $= 4$"
- "LR $= 5 \times 10^{-4}$, EP $= 3$"
- "LR $= 5 \times 10^{-4}$, EP $= 7$"

- "LR = 10^{-4}, EP = 4"
- "LR = 10^{-4} EP = 10"

The best model for each dataset and each method is chosen based on the final loss of the development set. We combine an actual batch size of 16 with 2 gradient accumulation steps per minibatch to artificially create a batch size of 32. The maximum sequence length is set to 512 for both input and output. For the decoding method, we use beam search with a beam size of 10.

5.3.1 Automatic Evaluation

Table 5.2 shows the automatic evaluation results on Car Manuals with BART and T5 as the baseline. **Hard** is our method using *Hard-Question2SRL* for *Question2SRL*. **Soft** is our method with *Soft-Question2SRL* with 80% as the soft-matching threshold (α). **Soft+C** and **Soft+L** are our methods with *Soft-Question2SRL* plus using the original answer or SRL labels and their corresponding words as the context, respectively.

As shown in Table 5.2, using SRL representations to train a QG model with either BART or T5 significantly improves the baseline performance on Car Manuals. This is due to generalization of sentences into SRL representations, allowing general semantic patterns to be modeled and used in QG. The use of SRLs also increases the number of training examples due to the fact that the SRLer can convert a sentence into multiple SRL-labeled sentences. All the variations of our method significantly increase the recall and F-scores in all types of measures (BLEU, ROUGE and METEOR) and they also increase precision in almost all of the cases. The recall increase is due to the fact that the SRLer can convert a sentence into multiple SRL-labeled sentences, leading to questions asking about different aspects of the sentence.

Table 5.2: Automatic evaluation results on **Car Manuals** with **BART** and **T5** as Baselines (P, R and F mean Precision, Recall and F-score in %). Hard and Soft+ methods are variations of our method with BART or T5 as the Seq2Seq model.

	BLEU-4			ROUGE-L			METEOR		
QG Method	F	P	R	F	P	R	F	P	R
BART	57.2	63.7	51.9	71.7	75.9	68.0	57.6	63.7	52.6
Hard	70.9	68.3	73.6	76.4	75.7	77.1	58.7	57.5	59.9
Soft	82.6	76.3	90.1	90.6	87.9	93.5	58.8	55.7	62.2
Soft+C	88.3	**85.4**	91.4	94.3	**94.0**	94.6	63.0	**62.1**	63.9
Soft+L	**89.0**	85.1	**93.2**	**94.8**	93.7	**95.9**	**63.6**	61.8	**65.5**
T5	45.0	50.3	40.7	62.4	66.0	59.2	46.3	50.0	43.1
Hard	63.9	58.8	70.0	71.5	69.1	74.0	52.3	49.3	55.7
Soft	77.0	71.5	83.5	85.9	82.7	89.4	53.6	50.4	57.1
Soft+C	**85.9**	**84.1**	**87.8**	**91.9**	**91.5**	**92.4**	**59.9**	**59.3**	**60.5**
Soft+L	84.7	82.8	86.6	91.0	90.1	92.0	58.8	57.6	60.0
Rule Based	17.4	13.9	23.2	36.2	31.6	42.5	22.0	18.5	27.1

Comparing the hard and soft versions of our method, Soft-Question2SRL is better than Hard-Question2SRL in all cases, indicating that allowing different but similar expressions in the corresponding question and answer when labeling the question with SRLs is important and beneficial. We also see that the use of the original source sentence or SRL labels as a context in the input is beneficial. This is probably because the error propagation from semantic role labeling has less impact when (part of) the original sentence is given as a context.

Tables 5.3 shows the automatic evaluation results on the SQuAD dataset with BART

Table 5.3: Automatic evaluation results on **SQuAD** with **BART** and **T5** as baselines (P, R and F mean Precision, Recall and F-score in %). Hard and Soft+ methods are different variations of our method with BART and T5 as the Seq2Seq models.

	BLEU-4			ROUGE-L			METEOR		
QG Method	F	P	R	F	P	R	F	P	R
BART	15.2	**21.4**	11.7	42.9	**48.0**	38.8	20.6	**22.6**	18.9
Hard	17.8	17.9	17.6	45.0	44.0	46.2	21.4	20.3	22.6
Soft	19.6	18.5	**20.9**	46.6	44.3	49.1	22.4	21.0	24.0
Soft+C	19.7	20.5	19.0	47.4	47.0	47.9	22.9	21.8	24.1
Soft+L	**20.0**	20.2	19.9	**48.1**	46.9	**49.3**	**23.3**	21.8	**25.0**
T5	15.0	19.9	12.0	41.0	46.3	36.7	19.5	**23.2**	16.7
Hard	16.6	16.1	17.1	43.4	42.2	44.5	20.3	19.6	21.1
Soft	18.0	16.6	**19.8**	44.8	43.0	46.8	21.5	20.9	22.2
Soft+C	19.0	19.6	18.4	45.7	45.7	45.8	22.1	22.2	22.0
Soft+L	**19.9**	**20.4**	19.4	**48.0**	**47.1**	**48.9**	**23.2**	21.8	**24.8**

and T5 as the baseline. Again, we observe that all variations of our method significantly outperform the baseline on recall and F-score. On this data set, the BART baseline shows the best precision. However, the lower precision of our method is due to the incompleteness of the ground truth questions in SQuAD. That is, many of the questions our method generates are good questions, but they do not match the ground truth questions in the data set (Such a problem is also mentioned by others such as in (Sultan et al., 2020)).

Table 5.4 shows the generated questions for 3 testing examples from our method (Soft+L) with T5 and T5 without SRL. As shown, our method generates more questions covering different aspects of an input sentence, while T5 without SRL generates only one question. In

Table 5.4: Three Examples showing the input sentence $\widehat{a}$, its ground-truth question q_g, the question generated by T5 (q_t), and the questions $\widehat{q}_i$ generated by our Soft+L method with T5 and alpha=80% on the SQuAD dataset.

$\widehat{a}$:	there are two categories of repetitive dna in genome : tandem repeats and interspersed repeats .
q_g:	what are two types of repetitive dna found in genomes ?
q_t:	what are the two categories of repetitive dna in genome ?
$\widehat{q}_1$:	what are the two categories of repetitive dna in the genome ?
$\widehat{q}_2$:	how many categories of repetitive dna are there in the genome ?
$\widehat{a}$:	before the solar/wind revolution , portugal had generated electricity from hydropower plants on its rivers for decades .
q_g:	through what renewable resource had portugal generated electricity before the solar/wind revolution ?
q_t:	before the solar/wind revolution, portugal had generated electricity from what ?
$\widehat{q}_1$:	what had portugal generated electricity from before the solar/wind revolution ?
$\widehat{q}_2$:	how long had portugal generated electricity from hydropower plants on its rivers ?
$\widehat{a}$:	the term parinirvana is also encountered in buddhism , and this generally refers to the complete nirvana attained by the arahant at the moment of death , when the physical body expires .
q_g:	what term is used for the complete nirvana attained by the arahant at death ?
q_t:	what is the term parinirvana used in buddhism ?
$\widehat{q}_1$:	what term is also encountered in buddhism ?
$\widehat{q}_2$:	when is the complete nirvana attained by the arahant ?
$\widehat{q}_3$:	who attained the complete nirvana at the moment of death ?
$\widehat{q}_4$:	what does the term parinirvana refer to at the moment of death ?
$\widehat{q}_5$:	what term is used in buddhism to describe the complete nirvana attained by the arahant at the moment of death ?

all the 3 examples, there is only one ground-truth question. When precision is computed, the extra questions we generated have a low precision due to poor match with the ground truth even though they are good questions. This explains why our methods have lower precision scores than the baselines.

The results on this dataset also show that soft matching is better than hard matching, and the use of context is better with T5, but has no obvious advantage for BART. We show the impacts of soft-matching threshold α in Section 5.5. In practice, α can be tuned using the development set.

Table 5.5: Automatic evaluation results on **NewsQA** with **BART** and **T5** as Baselines (P, R and F mean Precision, Recall and F-score in %). Hard and Soft+ methods are different variations of our method with BART and T5 as the Seq2Seq models.

	BLEU-4			ROUGE-L			METEOR		
QG Method	F	P	R	F	P	R	F	P	R
BART	10.7	**16.8**	7.8	38.3	**44.7**	33.6	17.6	**20.6**	15.3
Hard	12.2	13.2	11.3	41.4	41.8	41.1	19.0	18.1	20.0
Soft+A80	13.4	12.8	**14**	42.8	41.9	**43.8**	19.9	18.9	**21.1**
Soft+A80+C	**14.8**	16.3	13.6	**43.8**	44.6	43.0	**20.6**	20.4	20.8
T5	10.4	**15.8**	7.8	37.6	**44.1**	32.8	17.1	**20.7**	14.6
Hard	11.9	12.6	11.2	40.9	41	40.8	18.8	18.2	19.6
Soft+A80	13.3	12.9	**13.8**	42.6	41.6	43.7	19.8	18.8	**20.8**
Soft+A80+C	**14.1**	15.7	12.9	**42.9**	43.9	42.0	**20.0**	20.3	19.8
Rule Based	4.5	3.8	5.6	22.8	21.8	24.0	12.8	14.2	11.7

Similar results are observed on the NewsQA dataset, as shown in Table 5.5.

5.3.2 Human Evaluation

We randomly selected 50 input sentences from the SQuAD dataset and asked 5 English speakers to rate the quality of generated questions from each method in terms of *recall*, *clarity*, *Q&A relatedness* and *grammar* on a scale of 1-5 (Heilman and Smith, 2010) using the following criteria.

For **Recall**, the ratings are:

- 1= Bad: the generated questions do not cover any fact in the answer;
- 2= Unacceptable: the generated questions cover only a small portion of the facts in the answer;
- 3= Borderline: the generated questions cover around 50% of the facts in the answer;
- 4= Acceptable: the generated questions cover most of the facts in the answer; and
- 5= Good: the generated questions cover all the facts in the answer.

For **Clarity**, the ratings are:

- 1= Bad: the question is completely unclear in meaning or makes no sense;
- 2= Unacceptable: the question is mostly unclear;
- 3= Borderline: the question is between unacceptable and acceptable;
- 4= Acceptable: the question is clear and understandable, but the use of words can be improved; and
- 5= Good: the question has no problem. It is clear and simple and uses the right words.

For **Q&A Relatedness**, the ratings are:

- 1= Bad: the question is completely unrelated to the answer sentence it is generated from;

- 2= Unacceptable: the question is somewhat related to the answer sentence, but it cannot be answered by the answer sentence;
- 3= Borderline: the question can be partially answered by the answer sentence, but far from completely;
- 4= Acceptable: the question can be mostly answered by the answer sentence, although maybe not completely; and
- 5= Good: he question can be very well answered by the answer sentence.

For **Grammar**, the ratings are:

- 1= Bad: the grammar of the question is completely wrong;
- 2= Unacceptable: the question has major grammatical problems;
- 3= Borderline: the question has a grammatical error, which is between major and minor;
- 4= Acceptable: the question has only a minor grammatical problem; and
- 5= Good: the question is completely grammatically correct.

A precision score for each question is computed by averaging the scores for clarity, Q&A relatedness and grammar. An overall score (F score) for each test input sentence is computed as the harmonic mean of precision and recall scores.

Table 5.6 shows the averages of all the scores among human evaluators. The results show that our methods (BART+Soft+L and T5+Soft+L) are better in precision, recall and F-scores. In particular, they are significantly better than their corresponding baseline in recall and F-measure.

Table 5.6: Human evaluation results on 50 test sentences from SQuAD. Precision is the average of Clarity, Relatedness and Grammar scores. F-measure is the harmonic mean of precision and recall scores.

QG System	F-measure Score (1-5) ± stdev	Precision Score (1-5)	Recall Score (1-5)	Clarity Score (1-5)	Relatedness Score (1-5)	Grammar Score (1-5)
BART	3.64 ± 0.33	4.73	3.06	4.70	4.56	**4.94**
Soft+L (with BART)	**4.32 ± 0.28**	**4.77**	**4.16**	**4.75**	**4.61**	4.93
T5	3.63 ± 0.37	4.71	3.08	4.66	**4.63**	4.85
Soft+L (with T5)	**4.40 ± 0.29**	**4.73**	**4.25**	**4.72**	4.48	**4.90**

5.4 Comparison with other SOTA methods

We compare our method with additional SOTA baselines, most of which can generate multiple questions from an input sentence:

- **BART-multi** and **T5-multi**. BART (Lewis et al., 2020) or T5 (Raffel et al., 2020) model fine-tuned to generate multiple sequences given an input sequence by using $<sep>$ tokens to separate the ground-truth questions in the output part of each training example.
- **BART-divbeam** and **T5-divbeam**. BART or T5 model that uses decoding-based diverse beam search (Vijayakumar et al., 2016) to generate multiple questions. We use beam size of 6 with 3 diverse groups and diversity penalty of 0.4 all same as in (Z. Zhang and Zhu, 2021). We select 3 best questions for the evaluation because the average number of questions generated by our method per input sentence is 3, to make the comparison fair. In general, the more questions are generated, the lower the precision but better the recall.

- **ProphetNet-single** and **ProphetNet-multi**. ProphetNet (Qi et al., 2020) is another Transformer-based SOTA model. In ProphetNet-single, ProphetNet is fine-tuned to generate a single question. In ProphetNet-multi, it is fine-tuned to generate multiple sequences from an input sequence by using $<sep>$ tokens to separate the ground-truth questions.
- **MP-GSN** (Zhao et al., 2018) An LSTM-based Seq2Seq QG model. We use the sentence-level MP-GSN with the default setting in the implementation of MP-GSN in (Lee, 2019).
- **KPCNet** (Z. Zhang and Zhu, 2021) A QG method that uses keyword and diverse-beam search to generate multiple clarification questions. We report results for different versions of this method. KPCNet(beam) and KPCNet(divbeam) use classical and diverse beam search respectively as their generation method. KPCNet(sample) and KPCNet(cluster) generate questions based on 2 groups of selected keywords. Each variant produces 3 questions.
- **Rule-based method** with 75 rules based on semantic role labels to convert a sentence into questions. An example rule is "Replace [ARG1] with *what* if it appears at the beginning of the sentence".

Similar to the experiments in the last section, the hyper-parameters for all the Seq2Seq methods are determined through random search.

Table 5.7 shows the results of these SOTA methods on SQuAD compared to our method (Soft+L) with BART or T5 as the Seq2Seq model. As shown, our methods outperform all the baselines in F-score and recall. It is also the best in precision in all underlying metrics except for precision using METEOR where ProphetNet-single is the best and our methods are the second best.

Table 5.7: Comparison with SOTA models on **SQuAD**

QG Method	BLEU-4			ROUGE-L			METEOR		
	F	P	R	F	P	R	F	P	R
BART-multi	15.2	18.4	12.9	41.8	44.5	39.3	20.0	21.2	18.8
T5-multi	14.9	16.5	13.6	40.5	42.2	39.0	19.3	20.9	17.9
BART-divbeam	17.5	19.2	16.1	46.7	46.9	46.5	22.6	21.7	23.5
T5-divbeam	17.6	19.2	16.3	46.3	46.4	46.1	22.4	21.8	23.1
ProphetNet-single	16.2	20.3	13.5	43.0	46.4	40.2	21.2	**22.7**	19.8
ProphetNet-multi	14.5	17.7	12.3	39.6	43.7	36.2	19.8	21.3	18.5
MP-GSN	12.1	17.0	9.5	39.5	43.9	35.8	17.7	19.1	16.5
KPCNet(beam)	6.7	6.6	6.8	35.9	34.3	37.6	13.9	12.0	16.5
KPCNet(divbeam)	6.0	5.3	6.9	34.2	31.4	37.4	13.3	11.2	16.4
KPCNet(sample)	6.7	6.4	7.1	35.3	33.8	36.9	13.8	12.2	15.9
KPCNet(cluster)	6.9	6.2	7.8	35.9	33.6	38.5	13.9	11.9	16.7
Rule Based	9.1	8.0	10.4	25.1	23.0	27.5	15.0	14.5	15.4
Soft+L (Ours with BART)	**20.0**	20.2	**19.9**	**48.1**	46.9	**49.3**	**23.3**	21.8	**25.0**
Soft+L (Ours with T5)	19.9	**20.4**	19.4	48.0	**47.1**	48.9	23.2	21.8	24.8

Compared to Seq2Seq-multi, our methods outperform them significantly in all metrics, indicating using SRL to generate multiple questions is much more effective than using the $<sep>$ tokens in the output parts of the training sequences.

Our methods also outperform the diverse-beam-search-based methods for generating multiple questions, indicating different SRL representations of a sentence can more effectively lead to generation of diverse questions that cover different aspects of the input sentence.

KPCNet was chosen as a baseline because it uses both (diverse) beam search and keywords

for generating diverse questions, and its keyword predictor is trainable. But KPCNet was designed to generate clarification questions. We adapted the method by training it on SQuAD to generate questions that can be answered by the input sentence (because the ground-truth questions are answerable). Also, we do not use keyword filtering (which is an effort for removing answerable questions). We notice that its keyword predictor tuned on SQuAD produces repetitive keywords (e.g., year and type) almost for every input sentence. Thus, many of their generated questions do not match with the ground-truth questions, which leads to its poor performance on SQuAD.

Compared to the rule-based method, our method outperforms it significantly on all measures with big margins. This indicates that SRL-based Seq2Seq model better captures the relations between SRL representations of questions and those of answers than rules, which complicated "rules" can be learned by Seq2Seq models.

Table 5.8: Average number of different generated questions per source sentence by our method with BART and T5

QG Method	Avg. # of questions
BART+Soft	3.3
BART+Soft+C	2.8
BART+Soft+L	3.0
T5+Soft	3.3
T5+Soft+C	2.7
T5+Soft+L	3.0

Table 5.8 provides the average number of questions generated by different versions of our methods on the SQuAD dataset.

Table 5.9: Three Examples showing the input sentence $\widehat{a}$, its ground-truth question q_g, the question generated by T5-divbeam (q_{td}), the questions q_{kb} generated by KPCNet(beam), and the questions q_{kdb} generated by KPCNet(divbeam) on the SQuAD dataset

$\widehat{a}$:	there are two categories of repetitive dna in genome : tandem repeats and interspersed repeats .
q_g:	what are two types of repetitive dna found in genomes ?
q_{td}:	what are the two categories of repetitive dna in genome ?
	what are the two categories of repetitive dna in the genome ?
	how many categories of repetitive dna are in genome ?
q_{kb}:	what are three categories of coding dna ?
	how many categories are there ?
	are there three categories of coding dna found in dna ?
q_{kdb}:	what are three categories of coding dna ?
	polymophism are usually categories of what ?
	are there three categories of coding dna found ?
$\widehat{a}$:	before the solar/wind revolution , portugal had generated electricity from hydropower plants on its rivers for decades .
q_g:	through what renewable resource had portugal generated electricity before the solar/wind revolution ?
q_{td}:	before the solar/wind revolution, portugal had generated electricity from what ?
	before the solar/wind revolution, portugal generated electricity from what ?
	what was portugal's main source of electricity before the solar/wind revolution ?
q_{kb}:	in what year did the ottoman revolution take electricity ?
	in what year did the merger take electricity from its river ?
	what year did the empire take electricity ?
q_{kdb}:	in what year did the merger take electricity ?
	how much did the ottoman revolution take electricity from ?
	during what years did the merger take electricity from its river ?
$\widehat{a}$:	the term parinirvana is also encountered in buddhism , and this generally refers to the complete nirvana attained by the arahant at the moment of death , when the physical body expires .
q_g:	what term is used for the complete nirvana attained by the arahant at death ?
q_{td}:	what term is also encountered in buddhism ?
	what does parinirvana refer to ?
	what does parinirvana mean ?
q_{kb}:	what is another term for the heian ?
	what is the term for the term of pain ?
	what is the term for the term ?
q_{kdb}:	what is another term for the term " pregnancy " ?
	who is the term of a term ?
	the term of what is used to be found in hinduism ?

In Table 5.9, we show some sample questions generated by T5 with the diverse beam search (T5-divbeam) and KPCNet. The results of our method on the same testing examples are provided in Table 5.4.

5.5 Sensitivity Analysis

To see how the soft-matching threshold α affects the results, we experiment with different α values ranging from 70 to 95 and use the resulting datasets to fine-tune T5. In this experiment, we use the best configurations of the T5 model on the Car Manuals and SQuAD datasets. Six α values of 70, 75, 80, 85, 90, and 95 are selected for this purpose.

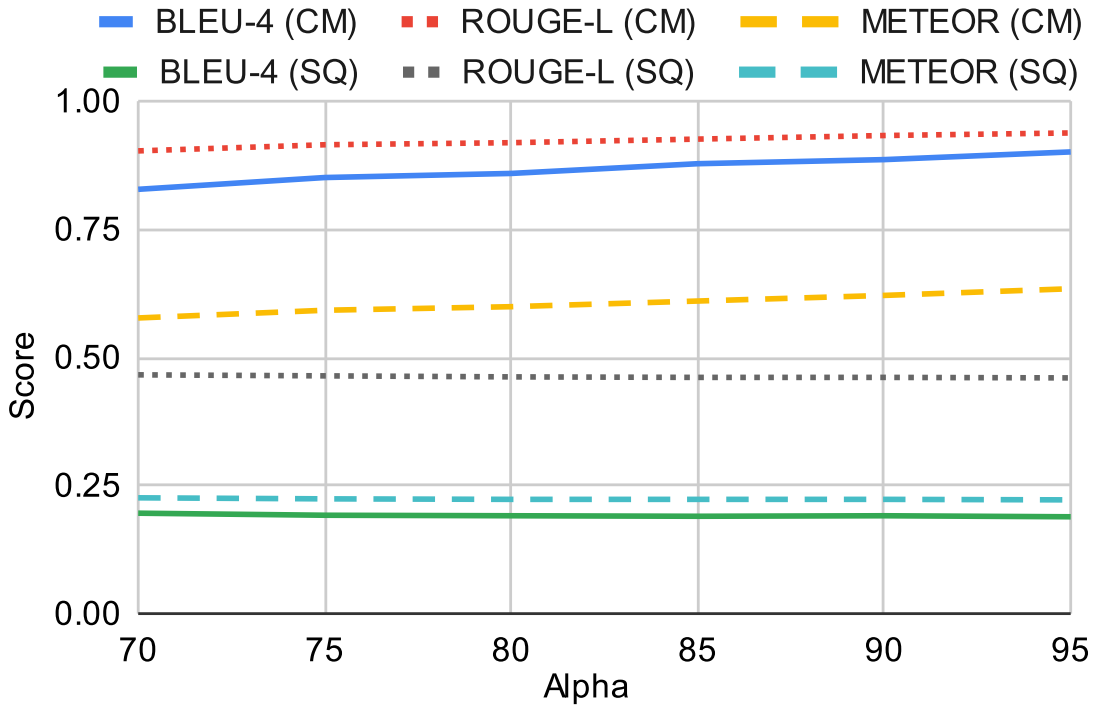


Figure 5.1: BLEU-4, ROUGE-L and METEOR F-scores of T5 on SQuAD (SQ) and Car Manuals (CM) using different alphas values.

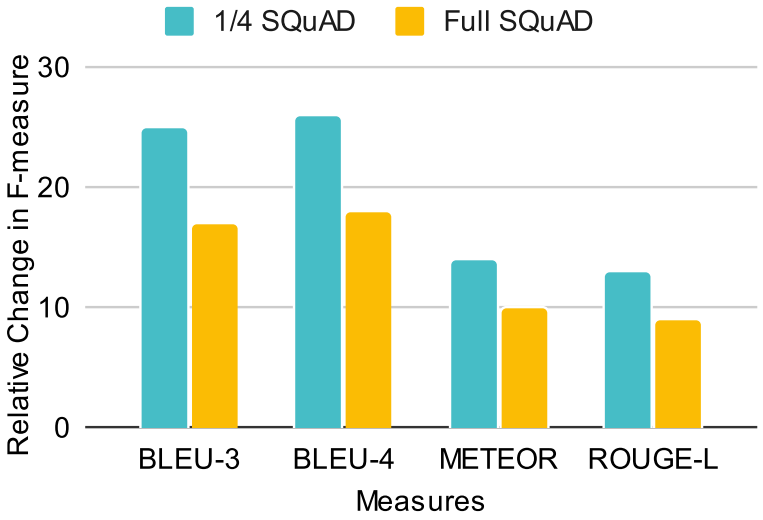
Figure 5.1 shows how the F-score of BLEU-4, ROUGE-L and METEOR changes with α

on SQuAD (SQ) and Car Manuals (CM). As can be seen, on SQuAD all the lines are quite flat, indicating α values do not have much impact on the performance. This is likely due to the best-matching n-gram in a ground-truth question either matches with the word/phrase replaced by an SRL in the answer very well or very poorly, leading to insensitivity to the threshold value between 70 and 95. Similar results are observed on the Car Manuals dataset although the scores are increasing with the α on this dataset, but slowly.

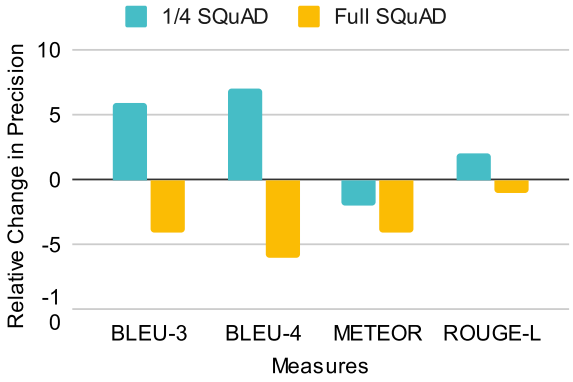
5.6 Performance with Different Data Sizes

Among the 3 datasets, Car Manuals is the smallest (with 9,184 training examples), and the improvements of our method on this dataset over the baselines are the largest. To further investigate whether our method makes better improvement when the data set is small, we conducted an experiment with one-quarter of the SQuAD dataset.

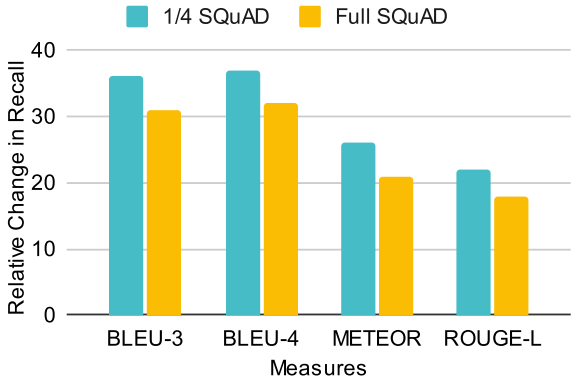
Figure 5.2 illustrates the relative improvement of our method over the T5 baseline in F-measure, precision and recall. The relative improvement is computed as: $\frac{score_{our} - score_{t5}}{score_{t5}}$. The blue bars in the figure represent the improvement on the one-quarter subset of SQuAD and the yellow ones represent that on the whole SQuAD dataset. Clearly, the improvements of our method over the T5 baseline are larger on the smaller dataset than on the whole SQuAD dataset. This indicates that our SRL-based Seq2Seq method can better handle smaller datasets than its Seq2Seq baselines due to the increase in training examples when we label the QAs with SRLs.



(a) F-measure



(b) Precision



(c) Recall

Figure 5.2: Comparison of relative change (percent) between the first quarter of SQuAD and whole SQuAD dataset using T5+C versus T5 with alpha=85%.

5.7 Evaluation of Paragraph-level Seq2Seq+SRL method

To discover whether the use of SRL with Seq2Seq models on datasets including long and complicated answers still leads to improvements over Seq2Seq models trained with original datasets, we performed experiments with the same setting described in section 5.3.

The automatic and human results reported in the following subsections are part of my 4-months internship at iNAGO Corporation.

5.7.1 Datasets

We evaluate our method on 2 datasets. The first one contains QAs created by human annotators from car manuals (of Ford, GM, Toyota, BMW, UC, and Tesla) and motorcycle manuals (of KTM) denoted as *Improved Manuals v1.4.1* provided by our industry partner. The second dataset is SQuAD (Rajpurkar et al., 2016), containing QAs created by Amazon Mechanical Turk crowd-workers from Wikipedia articles. We use the processed paragraph-level SQuAD dataset (Du et al., 2017), where the answer in a QA pair is a paragraph.

Table 5.10: Statistics of Two Paragraph-level Datasets

Dataset	training set	test set	development set
Improved Manuals v1.4.1	21,064 QAs	1,367 QAs	4,067 QAs
SQuAD	70,484 QAs	11,877 QAs	10,570 QAs

Table 5.10 gives the statistics of aforementioned datasets. In these two datasets, multiple QA pairs may have the same answer paragraph but different questions.

5.7.2 Automatic Evaluation

Table 5.11 shows the automatic evaluation results on Improved Manuals v1.4.1 with BART and T5 as the baseline. **Soft+C** is our method with *Soft-Question2SRL* after sentence segmentation plus using the original paragraph as the context with 80% as the soft-matching threshold (α). **Q&A Generator** is iNAGO's previous generator fine-tuned on Improved Manuals v1.4.1 with T5-base model. Note that we use the smallest available model size of T5 which is T5-small.

Table 5.11: Automatic evaluation results on **Improved Manuals v1.4.1** with **BART** and **T5** as Baselines (P, R and F mean Precision, Recall and F-score in %). Soft+C is our method plus using the original paragraph as the context with BART or T5 as the Seq2Seq model.

	BLEU-4			ROUGE-L			METEOR		
QG Method	F	P	R	F	P	R	F	P	R
BART	40.3	49.4	34.1	57.1	64.7	51.2	35.8	39.2	33.0
Soft+C	**53.5**	**53.9**	**53.0**	**71.4**	**72.9**	**70.1**	**41.4**	**41.6**	**41.2**
T5	40.8	48.5	35.2	56.9	64.6	50.9	35.5	40.2	31.8
Soft+C	**51.5**	**53.7**	**49.4**	**69.7**	**71.6**	**68.0**	**40.3**	**41.4**	**39.3**
Q&A Generator	39.2	48.3	33.1	56.3	63.7	50.5	35.2	38.7	32.3

As shown in Table 5.11, using SRL representations to train a paragraph-level QG model with either BART or T5 considerably improves the baseline performance on Improved Manuals v1.4.1. Our method significantly increases the precision, recall and F-scores in all types of measures (BLEU, ROUGE and METEOR). The percentage increase in precision, recall and F-measure achieved by this method over the baseline ranges 3%-13%, 24%-55% and 14%-33%, respectively.

Both baselines (BART and T5) outperform Q&A Generator gently in all metrics even

though it is a larger model based on T5-base with 223 million parameters. To bear in mind BART-base and T5-small have 139 and 60 million parameters respectively. It shows random search (Bergstra and Bengio, 2012) over the same domain is able to find better hyperparameters leading to better results even with smaller models.

Table 5.12: Automatic evaluation results on **SQuAD** with **BART** and **T5** as Baselines (P, R and F mean Precision, Recall and F-score in %). Soft+C is our method plus using the original paragraph as the context with BART or T5 as the Seq2Seq model.

	BLEU-4			ROUGE-L			METEOR		
QG Method	F	P	R	F	P	R	F	P	R
BART	11.5	**28.6**	7.2	39.9	**53.6**	31.8	18.2	**25.8**	14.1
Soft+C	**20.6**	27.0	**16.7**	**48.4**	51.6	**45.7**	**23.3**	24.4	**22.3**
T5	11.0	**24.3**	7.1	37.3	**51.0**	29.4	16.5	**26.1**	12.1
Soft+C	**19.5**	24.0	**16.3**	**46.2**	50.0	**42.9**	**22.1**	25.2	**19.8**

The automatic evaluation results on the SQuAD dataset with BART and T5 as the baseline are given in Table 5.12. We observe that our method significantly outperforms the baseline on recall and F-score on this dataset. However, our method has lower precision compared to the baseline owing to the fact that the ground truth questions in SQuAD are incomplete. As explained in Section 5.3.1, many of the questions generated by our method are good, but they do not match the ground truth questions in the dataset (Sultan et al., 2020).

Table 5.13: Three Examples showing the input paragraph $\widehat{a}$, its ground-truth question q_g, the question generated by BART (q_b), and the questions $\widehat{q}_i$ generated by our Soft+C method with BART and alpha=80% on the SQuAD dataset.

$\widehat{a}$:	over time , electric lighting became ubiquitous in developed countries . segmented sleep patterns disappeared , improved nighttime lighting made more activities possible at night , and more street lights reduced urban crime .
q_g:	street lights help reduce ?
q_b:	what replaced segmented sleep patterns ?
$\widehat{q}_1$:	what became ubiquitous in developed countries ?
$\widehat{q}_2$:	how did more street lights reduced urban crime ?
$\widehat{q}_3$:	when did electric lighting become ubiquitous in developed countries ?
$\widehat{a}$:	schwarzenegger has been a registered republican for many years . as an actor , his political views were always well known as they contrasted with those of many other prominent hollywood stars , who are generally considered to be a liberal and democratic-leaning community . at the 2004 republican national convention , schwarzenegger gave a speech and explained why he was a republican .
q_g:	in what year did schwarzenegger speak at the republican national convention ?
q_b:	what political party does schwarzenegger belong to ?
$\widehat{q}_1$:	how long has schwarzenegger been a republican ?
$\widehat{q}_2$:	what political party did schwarzenegger belong to ?
$\widehat{q}_3$:	when did schwarzenegger speak at the republican national convention ?
$\widehat{a}$:	schwarzenegger is considered among the most important figures in the history of bodybuilding , and his legacy is commemorated in the arnold classic annual bodybuilding competition . schwarzenegger has remained a prominent face in the bodybuilding sport long after his retirement , in part because of his ownership of gyms and fitness magazines . he has presided over numerous contests and awards shows .
q_g:	what bodybuilding competition is named after schwarzenegger ?
q_b:	who is considered among the most important figures in the history of bodybuilding ?
$\widehat{q}_1$:	when did schwarzenegger retire ?
$\widehat{q}_2$:	what is schwarzenegger's legacy commemorated in ?
$\widehat{q}_3$:	what is the arnold classic annual bodybuilding competition ?
$\widehat{q}_4$:	how many contests and awards shows has schwarzenegger presided over ?
$\widehat{q}_5$:	who is considered among the most important figures in the history of bodybuilding ?

Table 5.13 provides the generated questions for 3 testing examples from our method (Soft+C) with BART and BART without SRL. As shown, our method generates more questions covering different aspects of an input paragraph, while BART without SRL generates only one question. In all the 3 examples, there is only one ground-truth question. When precision is computed, the extra questions we generated have a low precision due to poor match with the ground truth even though they are good questions. It was for that reason that our methods have lower precision scores than the baselines.

5.7.3 Human Evaluation

The Content Team at iNAGO feed 95 randomly selected input paragraphs from different chapters of car and motorcycle manuals (which means they are not included in Improved Manuals v1.4.1 but have the same distribution) to Q&A Generator and Soft+C (with BART). Two English speaker linguists rated the quality of generated questions from each method in terms of *recall*, *conciseness*, *structure*, *meaning* and *Q&A relatedness* using the following criteria.

For **Recall**, the ratings are:

- 1: The generated questions cover no fact or a small portion of the facts in the answer;
- 2: The generated questions cover around 50% of the facts in the answer;
- 3: The generated questions cover most of or all the facts in the answer.

For **Conciseness**, the ratings are:

- 1: The generated questions are vague and miss some information in the answer;
- 2: The generated questions are not concise but otherwise convey all information in the answer;

- 3: The generated questions are clear and simple, and convey all information in the answer.

For **Structure**, the ratings are:

- 1: The generated questions have several major or minor grammatical errors (e.g., missing words, gibberish characters, bad sentence structure) that severely affect the intent;
- 2: The generated questions have a few minor grammatical syntax errors (e.g., spelling, switched words, incorrect punctuation) that could easily be fixed and do not otherwise affect the intent;
- 3: The generated questions have no grammatical errors.

For **Meaning**, the ratings are:

- 1: The generated questions have a mismatch (e.g., can I close the engine?) between the words being used;
- 2: The generated questions have words not used in a completely correct way but are understandable;
- 3: The generated questions use the right word and completely make sense.

For **Q&A Relatedness**, the ratings are:

- 1: The generated questions contain relevant keywords/concepts (e.g., features, states, parts) but are unrelated context to the answer;
- 2: The generated questions contain relevant keywords/concepts in a similar context to the answer;
- 3: The generated questions are related to the answer.

Table 5.14: Human evaluation results on 95 input paragraphs with the same distribution as in **Improved Manuals v1.4.1**. Precision is the average of Conciseness, Structure, Meaning, and Relatedness scores. F-measure is the harmonic mean of precision and recall scores.

QG System	F-measure Score (1-3)	Precision Score (1-3)	Recall Score (1-3)	Conciseness Score (1-3)	Structure Score (1-3)	Meaning Score (1-3)	Relatedness Score (1-3)
Q&A Generator	2.08	2.44	1.92	2.53	**2.66**	2.53	2.04
Soft+C (with BART)	**2.36**	**2.57**	**2.28**	**2.72**	2.63	**2.56**	**2.39**

Table 5.14 shows the averages of all the scores among two linguists on a scale of 1-3, and the average score of the generated questions having each criteria for each method. The precision score is the average among conciseness, structure, meaning and Q&A relatedness. F-measure is the harmonic mean of precision and recall scores. The results show that our method (BART+Soft+C) is significantly better in precision, recall and F-scores than the Q&A Generator. It is worth noting that the Q&A Generator generates two questions for each input paragraph using beam search with a beam size of 4.

Chapter 6

Conclusions

6.1 Summary of the thesis

We proposed a novel QG method that learns a Seq2Seq model to convert an SRL representation of an input sentence into an SRL representation of a question, which is then converted to a natural language question. Similar to rule-based methods, our SRL-based Seq2Seq methods can generate multiple questions from an input sentence, significantly improving the recall and overall performance of Seq2Seq QG. It is also much better than rule-based methods because better and more complicated "rules" can be learned via the Seq2Seq model. Our evaluation on three real-world datasets shows that the proposed method significantly outperforms both rule-based, original Seq2Seq methods and several other SOTA models, especially in recall and overall performance.

In an effort to extend our SRL-based method to paragraph-level QG, we apply our proposed sentence-level QG method after sentence segmentation of an input paragraph. In this way, a Seq2Seq model is fine-tuned to map an SRL representation of an input sentence along with its original paragraph into an SRL representation of a question, which is later

transformed into a natural language question. As a result of both automatic and human evaluations, we observe significant improvements compared to the original Seq2Seq methods on two real-world datasets.

Both our sentence-level and paragraph-level methods offer better generalization (trained with more generalized SRL labeled data) than Seq2Seq method which requires a large data set to generalize well. Thus, the proposed method provides an effective way to deal with limited training data.

6.2 Limitations and Future Work

A limitation of our method is that its performance depends on the SRL performance. If the input sentences are not well-formed (e.g., not grammatically correct, as occurs often in short social media messages, such as tweets), semantic role labeling may not produce correct labels. Also, we found that some of the questions generated by our method for the same input sentence may be similar or same in meaning with some minor differences in the use of words. This may not be a problem if the application allows similar questions to be generated (e.g., for reading comprehension). But in the application where duplicated questions are not allowed or not desired, a post-processing step to remove questions with the same meaning is needed. In addition, a grammatical correction module can be added as an extra post-processing step in order to improve the quality of generated questions with minor grammar issues. Lastly, sensitivity analysis, as discussed in Section 5.5, showed that the result does not depend on the alpha value greatly. But in practice, it is better to select the best alpha value using the development set, which we leave for future work.

Bibliography

Bengio, Yoshua, Patrice Simard, and Paolo Frasconi (1994). "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks* 5.2, pp. 157–166.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu (July 2002). "Bleu: a Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, pp. 311–318.

Bird, Steven and Edward Loper (July 2004). "NLTK: The Natural Language Toolkit". In: *Proceedings of the ACL Interactive Poster and Demonstration Sessions*. Barcelona, Spain: Association for Computational Linguistics, pp. 214–217. URL: https://aclanthology.org/P04-3031.

Lin, Chin-Yew (Jan. 2004). "ROUGE: A Package for Automatic Evaluation of summaries". In: p. 10.

Palmer, Martha, Daniel Gildea, and Paul Kingsbury (2005). "The proposition bank: An annotated corpus of semantic roles". In: *Computational linguistics* 31.1, pp. 71–106.

Punyakanok, Vasin, Dan Roth, and Wen-tau Yih (2008). "The Importance of Syntactic Parsing and Inference in Semantic Role Labeling". In: *Computational Linguistics* 34.2, pp. 257–287. DOI: 10.1162/coli.2008.34.2.257. URL: https://aclanthology.org/J08-2005.

Heilman, Michael and Noah A. Smith (2010). "Rating Computer-generated Questions with Mechanical Turk". In: *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*. CSLDAMT '10. Los Angeles, California: Association for Computational Linguistics, pp. 35–40. URL: http://dl.acm.org/citation.cfm?id=1866696.1866701.

Choi, Jinho D. and Martha Palmer (2011). "Transition-Based Semantic Role Labeling Using Predicate Argument Clustering". In: *Proceedings of the ACL 2011 Workshop on Relational Models of Semantics*. RELMS '11. Portland, Oregon: Association for Computational Linguistics, pp. 37–45. ISBN: 9781932432985.

Collobert, Ronan, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa (2011). "Natural language processing (almost) from scratch". In: *Journal of machine learning research* 12.ARTICLE, pp. 2493–2537.

Bergstra, James and Yoshua Bengio (2012). "Random search for hyper-parameter optimization." In: *Journal of machine learning research* 13.2.

Lindberg, David, Fred Popowich, John Nesbit, and Phil Winne (2013). "Generating natural language questions to support learning on-line". In: *Proceedings of the 14th European Workshop on Natural Language Generation*, pp. 105–114.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean (2013). "Distributed Representations of Words and Phrases and Their Compositionality". In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'13. Lake Tahoe, Nevada: Curran Associates Inc., pp. 3111–3119.

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio (2013). "On the difficulty of training recurrent neural networks". In: *International conference on machine learning*. PMLR, pp. 1310–1318.

Cho, Kyunghyun, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (Oct. 2014). "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 1724–1734. DOI: 10.3115/v1/D14-1179. URL: https://aclanthology.org/D14-1179.

Denkowski, Michael and Alon Lavie (2014). "Meteor Universal: Language Specific Translation Evaluation for Any Target Language". In: *In Proceedings of the Ninth Workshop on Statistical Machine Translation*.

Pennington, Jeffrey, Richard Socher, and Christopher Manning (Oct. 2014). "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://aclanthology.org/D14-1162.

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le (2014). "Sequence to Sequence Learning with Neural Networks". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc., pp. 3104–3112. URL: http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

Chali, Yllias and Sadid A. Hasan (Mar. 2015). "Towards Topic-to-Question Generation". In: *Computational Linguistics* 41.1, pp. 1–20. DOI: 10.1162/COLI_a_00206. URL: https://www.aclweb.org/anthology/J15-1001.

Khan, Atif, Naomie Salim, and Yogan Jaya Kumar (2015). "A framework for multi-document abstractive summarization based on semantic role labelling". In: *Applied Soft Computing* 30, pp. 737–747.

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E Hinton (2016). "Layer normalization". In: *arXiv preprint arXiv:1607.06450*.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Rajpurkar, Pranav, Jian Zhang, Konstantin Lopyrev, and Percy Liang (Nov. 2016). "SQuAD: 100,000+ Questions for Machine Comprehension of Text". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, pp. 2383–2392. DOI: 10.18653/v1/D16-1264. URL: https://www.aclweb.org/anthology/D16-1264.

Trischler, Adam, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman (2016). "Newsqa: A machine comprehension dataset". In: *arXiv preprint arXiv:1611.09830*.

Vijayakumar, Ashwin K., Michael Cogswell, Ramprasaath R. Selvaraju, Qing Sun, Stefan Lee, David J. Crandall, and Dhruv Batra (2016). "Diverse Beam Search: Decoding Diverse Solutions from Neural Sequence Models". In: *ArXiv* abs/1610.02424.

Du, Xinya, Junru Shao, and Claire Cardie (2017). "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)*, pp. 1342–1352.

Gardner, Matt, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer (2017). "AllenNLP: A Deep Semantic Natural Language Processing Platform". In: eprint: arXiv:1803.07640.

Joulin, Armand, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov (Apr. 2017). "Bag of Tricks for Efficient Text Classification". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics, pp. 427–431. URL: https://aclanthology.org/E17-2068.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin (2017). "Attention Is All You Need". In: *CoRR* abs/1706.03762. arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

Wang, Wenhui, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou (July 2017). "Gated Self-Matching Networks for Reading Comprehension and Question Answering". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, pp. 189–198. DOI: 10.18653/v1/P17-1018. URL: https://www.aclweb.org/anthology/P17-1018.

Yuan, Xingdi, Tong Wang, Caglar Gulcehre, Alessandro Sordoni, Philip Bachman, Sandeep Subramanian, Saizheng Zhang, and Adam Trischler (2017). "Machine comprehension by text-to-text neural question generation". In: *arXiv preprint arXiv:1705.02012*.

Chen, Guanliang, Jie Yang, Claudia Hauff, and Geert-Jan Houben (2018). "LearningQ: a large-scale dataset for educational question generation". In: *Twelfth International AAAI Conference on Web and Social Media*.

Flor, Michael and Brian Riordan (June 2018). "A Semantic Role-based Approach to Open-Domain Automatic Question Generation". In: *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 254–263. DOI: 10.18653/v1/W18-0530. URL: https://www.aclweb.org/anthology/W18-0530.

Khullar, Payal, Konigari Rachna, Mukul Hase, and Manish Shrivastava (2018). "Automatic question generation using relative pronouns and adverbs". In: *Proceedings of ACL 2018, Student Research Workshop*, pp. 153–158.

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

Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer (June 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)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 2227–2237. DOI: 10.18653/v1/N18-1202. URL: https://aclanthology.org/N18-1202.

Ren, Gary, Xiaochuan Ni, Manish Malik, and Qifa Ke (2018). "Conversational query understanding using sequence to sequence modeling". In: *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, pp. 1715–1724.

Song, Linfeng, Zhiguo Wang, Wael Hamza, Yue Zhang, and Daniel Gildea (June 2018). "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)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 569–574. DOI: 10.18653/v1/N18-2090. URL: https://aclanthology.org/N18-2090.

Subramanian, Sandeep, Tong Wang, Xingdi Yuan, Saizheng Zhang, Adam Trischler, and Yoshua Bengio (July 2018). "Neural Models for Key Phrase Extraction and Question Generation". In: *Proceedings of the Workshop on Machine Reading for Question Answering*.

Melbourne, Australia: Association for Computational Linguistics, pp. 78–88. DOI: 10.18653/v1/W18-2609. URL: https://aclanthology.org/W18-2609.

Sun, Xingwu, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma, and Shi Wang (Oct. 2018). "Answer-focused and Position-aware Neural Question Generation". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, pp. 3930–3939. DOI: 10.18653/v1/D18-1427. URL: https://aclanthology.org/D18-1427.

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

Yao, Kaichun, Libo Zhang, Tiejian Luo, Lili Tao, and Yanjun Wu (2018). "Teaching Machines to Ask Questions." In: *IJCAI*, pp. 4546–4552.

Zhao, Yao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke (Oct. 2018). "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*. Brussels, Belgium: Association for Computational Linguistics, pp. 3901–3910. DOI: 10.18653/v1/D18-1424. URL: https://aclanthology.org/D18-1424.

Zhou, Qingyu, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou (Jan. 2018). "Neural Question Generation from Text: A Preliminary Study". In: *Proceedings of the National CCF Conference on Natural Language Processing and Chinese Computing*. Springer, pp. 662–671.

Dong, Li, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon (2019). "Unified Language Model Pre-training for Natural Language Understanding and Generation". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and

R. Garnett. Vol. 32. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2019/file/c20bb2d9a50d5ac1f713f8b34d9aac5a-Paper.pdf.

Lee, Seanie (2019). "Pytorch implmentation of Paragraph-level Neural Question Generation with Maxout Pointer and Gated Self-attention Networks". In: URL: %5Curl%7Bhttps://github.com/seanie12/neural-question-generation%7D.

Li, Zuchao, Shexia He, Hai Zhao, Yiqing Zhang, Zhuosheng Zhang, Xi Zhou, and Xiang Zhou (2019). "Dependency or span, end-to-end uniform semantic role labeling". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01, pp. 6730–6737.

Reimers, Nils and Iryna Gurevych (2019). "Sentence-bert: Sentence embeddings using siamese bert-networks". In: *arXiv preprint arXiv:1908.10084*.

Shi, Peng and Jimmy Lin (2019). "Simple BERT Models for Relation Extraction and Semantic Role Labeling". In: *CoRR* abs/1904.05255. arXiv: 1904.05255. URL: http://arxiv.org/abs/1904.05255.

Song, Linfeng, Daniel Gildea, Yue Zhang, Zhiguo Wang, and Jinsong Su (2019). "Semantic Neural Machine Translation Using AMR". In: *Transactions of the Association for Computational Linguistics* 7, pp. 19–31. DOI: 10.1162/tacl_a_00252. URL: https://aclanthology.org/Q19-1002.

Wolf, Thomas, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. (2019). "Huggingface's transformers: State-of-the-art natural language processing". In: *arXiv preprint arXiv:1910.03771*.

Yuan, Xingdi, Tong Wang, Adam Peter Trischler, and Sandeep Subramanian (Feb. 2019). *Neural models for key phrase detection and question generation*. US Patent App. 15/667,911.

Bao, Hangbo, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Jianfeng Gao, Songhao Piao, Ming Zhou, and Hsiao-Wuen Hon (13–18 Jul 2020). "UniLMv2: Pseudo-

Masked Language Models for Unified Language Model Pre-Training". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 642–652. URL: https://proceedings.mlr.press/v119/bao20a.html.

Chen, Shizhe, Yida Zhao, Qin Jin, and Qi Wu (2020). "Fine-grained video-text retrieval with hierarchical graph reasoning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10638–10647.

Lewis, Mike, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer (July 2020). "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 7871–7880. DOI: 10.18653/v1/2020.acl-main.703. URL: https://aclanthology.org/2020.acl-main.703.

Pan, Youcheng, Baotian Hu, Qingcai Chen, Yang Xiang, and Xiaolong Wang (2020). "Learning to Generate Diverse Questions from Keywords". In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8224–8228. DOI: 10.1109/ICASSP40776.2020.9053822.

Qi, Weizhen, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou (Nov. 2020). "ProphetNet: Predicting Future N-gram for Sequence-to-SequencePre-training". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, pp. 2401–2410. DOI: 10.18653/v1/2020.findings-emnlp.217. URL: https://aclanthology.org/2020.findings-emnlp.217.

Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu (2020). "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *Journal of Machine Learning Research* 21.140, pp. 1–67. URL: http://jmlr.org/papers/v21/20-074.html.

Schlichtkrull, Michael Sejr and Weiwei Cheng (2020). "Evaluating for diversity in question generation over text". In: *arXiv preprint arXiv:2008.07291*.

Shen, Sheng, Yaliang Li, Nan Du, X. Wu, Yusheng Xie, Shen Ge, Tao Yang, Kai Wang, Xingzheng Liang, and Wei Fan (2020). "On the Generation of Medical Question-Answer Pairs". In: *AAAI*.

Sultan, Md Arafat, Shubham Chandel, Ramón Fernandez Astudillo, and Vittorio Castelli (July 2020). "On the Importance of Diversity in Question Generation for QA". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 5651–5656. DOI: 10.18653/v1/2020.acl-main.500. URL: https://aclanthology.org/2020.acl-main.500.

Xiao, Dongling, Han Zhang, Yukun Li, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang (July 2020). "ERNIE-GEN: An Enhanced Multi-Flow Pre-training and Fine-tuning Framework for Natural Language Generation". In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*. Ed. by Christian Bessiere. Main track. International Joint Conferences on Artificial Intelligence Organization, pp. 3997–4003. DOI: 10.24963/ijcai.2020/553. URL: https://doi.org/10.24963/ijcai.2020/553.

Gu, Jing, Mostafa Mirshekari, Zhou Yu, and Aaron Sisto (Apr. 2021). "ChainCQG: Flow-Aware Conversational Question Generation". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, pp. 2061–2070. DOI: 10.18653/v1/2021.eacl-main.177. URL: https://aclanthology.org/2021.eacl-main.177.

Lin, Juei-Yian, Jhih-Yuan Huang, and Wei-Po Lee (2021). "Question-Answer Generation for Data Augmentation". In: *2021 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing Communications (GreenCom) and IEEE Cyber, Physical Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, pp. 391–397. DOI: 10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics53846.2021.00069.

Liu, Jian, Yufeng Chen, and Jinan Xu (Nov. 2021). "Machine Reading Comprehension as Data Augmentation: A Case Study on Implicit Event Argument Extraction". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, pp. 2716–2725. DOI: 10.18653/v1/2021.emnlp-main.214. URL: https://aclanthology.org/2021.emnlp-main.214.

Pyatkin, Valentina, Paul Roit, Julian Michael, Yoav Goldberg, Reut Tsarfaty, and Ido Dagan (Nov. 2021). "Asking It All: Generating Contextualized Questions for any Semantic Role". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, pp. 1429–1441. DOI: 10.18653/v1/2021.emnlp-main.108. URL: https://aclanthology.org/2021.emnlp-main.108.

Zhang, Zhiling and Kenny Q. Zhu (2021). "Diverse and Specific Clarification Question Generation with Keywords". In: *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*. Ed. by Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia. ACM / IW3C2, pp. 3501–3511. DOI: 10.1145/3442381.3449876. URL: https://doi.org/10.1145/3442381.3449876.

Papay, Sean, Roman Klinger, and Sebastian Pado (2022). "Constraining Linear-chain CRFs to Regular Languages". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=jbrgwbv8nD.

Zhang, Jingqing, Yao Zhao, Mohammad Saleh, and Peter J. Liu (2022). "PEGASUS: Pre-Training with Extracted Gap-Sentences for Abstractive Summarization". In: *Proceedings of the 37th International Conference on Machine Learning*. ICML'20. JMLR.org.