Cue Word Guided Question Generation with BERT Model Fine-Tuned on Natural Question Dataset

A thesis submitted in partial fulfillment
of the requirements for
Master of Science (Research)
in
Department of Computer Science
at
The University of Waikato
by
Zijing Zhang
supervised by
Dr. Tony Smith

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

August 28, 2020

Abstract

This thesis aims to develop an efficient question generator for automated tutoring system. Given a context passage with an answer, the question generator asks questions to help the reader learn new material. By utilizing the BERT model, this thesis experiments on generating type-specific questions with a cue word. This thesis also uses an RNN encoder-decoder architecture for question generation on SQuAD as a comparative baseline and fine-tune the BERT question generation model on Google's Natural Question dataset. Ultimately, I deliver a RESTful API by the end of this year-long master program.

1 Introduction

Machine comprehension is the understanding of a query and its context to perform complex interactions between them. [30] Research in this domain aims to teach machines to understand contextual material such as text or pictures like a human would. SQuAD, CNN/Daily Mail, CBT, NewsQA, TriviaQA, WIKIHOP, and Google's Natural Question altogether give enough datasets to train neural machine comprehension models. [44] The most rewarding task now lies in using what machines have learned to solve urgent issues. There are two ways to access the knowledge from the machine to perform such a task: making them answer questions or, vice versa, letting them ask meaningful questions.

Question Generation Asking questions is a direct way to access knowledge. A good question is as impressive as a good answer for language comprehension. Given a context passage and an answer cue, the question generator creates a sequence of question tokens. In a seq-to-seq approach, a model first conditions on the context with the answer, and its starting token position. After this, a decoder sequentially chooses a word label ID from a tokenizer vocabulary as the predicted question. Automatic question generation is of interest from Microsoft[42] and Baidu Inc[40] for data augmentation in the question answering task. This paper proposes a starting token for generating questions to fit the need for making the question of a specific type. We first examine the prerequisite knowledge and prior works in question generation task. After which we experiment on datasets from the SQuAD and the natural question. Then, we explore a cue word impact on the question generated.

Application Data augmentation constructs new samples or algorithm for the model optimization. [36] Tang, Duan etc. from Microsoft Research Asia and Beihang University studied question generation and question answering as a joint task where by they optimized the question answering model with question generation task.[35] Heilman and Smith, 2010; Ali et al., 2010; Lindberg et al., 2013; Labutov et al., 2015; Mazidi and Nielsen, 2015 are using question generation as an automated solution for educational purpose. Zhao et al., 2011 implemented an question generation based rerouting algorithm to direct customer query to a Q.A. web page.

Why Neural? The neural network is good at capturing "feels" or fuzzy logic, while symbolic computation is good at expressing the rules and quick replication. Which school to follow is out of the scope of this work. It should be up to the tasks at hand to determine which methodology to deploy on computation. The question generation is fuzzy. The rule-based system often lacks high-level abstraction and renders the question generated trivially. Early development of deep learning approaches is often misleading due to its simplicity in architecture or the indescribable nature of computation. As the neural network gains popularity and promising result, its shroud of doubt fades with experiments. Deep learning models can self-organize a feature hierarchy with automated feature recognition, while conventional methods demand expensive expert hand-labeling. [22]

Scope of Work Symbolic computation is insufficient in low level or hard tasks such as question generation, similar to a Python Django web application programmer would get frustrated with the compiler algorithm or methods to handle assembly code. It's not a matter of whether assembly language like MIPS is superior in all aspect than a high level Python for coding. Rather, it's a matter of different use cases. With addition as in 1+1=2, its neural network counterparts would give 1.999 repeating with the highest amount of computational accuracy. But it would be wrong for such basic algorithmic operation in algebra. This work, in a metaphor, should be more of an experiment in creating a 1.999 for addition, rather than claiming that we have solved the mystery for addition operation or algebra. A human's inquisitive mind may expand far beyond algebra. Similar entities and verbs are often swapped in the

questions generated. For example, Miami is often swapped with Jacksonville. [42] Syntactical parsing an answer into a question most often carries few interests for reading comprehension. A meaningful question needs the right amount of abstraction and the right amount of detail. This high-level balancing is hard to achieve.[42] Lacking "common sense" is the most apparent drawback of models deployed to generate questions. [42] For instance, a question generator would treat the word "Audi" as an isolated entity, instead of relating it to cars, unless explicitly trained with such example. An overlapping metric counts the amount of character overlaps between candidate string and reference sentence. Widely used overlapping metric such as BLEU score is insufficient in measuring question generation quality since there are multiple ground truth or valid question to a given context. [42]

2 Prior Work

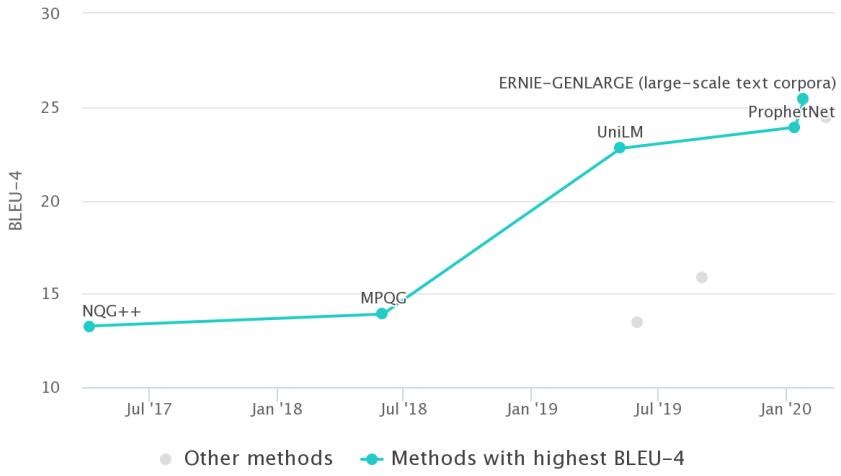

Figure 1: Question Generation Leaderboard: state of the art question researches that are present in the website paparwithcodes.com, fetched at August 25th, 2020

Question generation is an ongoing topic with high competition (as shown in Figure 1). This section examines question generation models that use rule-based systems, RNN with LSTM, and the BERT language model. The Neural Question Generation model is the first question generation RNN model with a

BLEU score around 15. ProphetNet and UniLM are from Microsoft as the best question generation models of their time. The latest model, ERNIE, is the best performing question generation model with a BLEU score of 25.

2.1 Hand-crafted Rule

2.1.1 Parse-Tree Based Question Generation

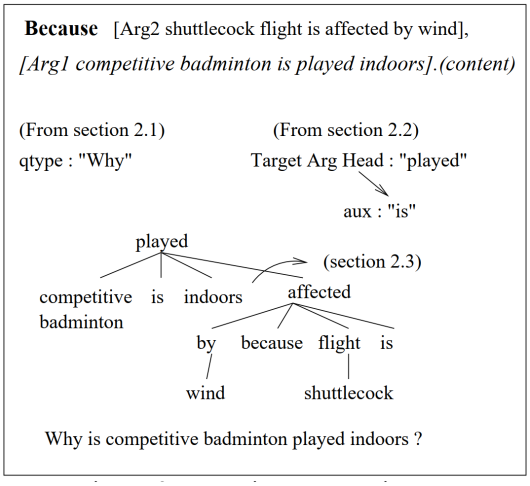

Figure 2: Discourse Connectives Question Generation note: the sections refer to original paper sections [2]

Discourse connectives	Sense	Q-type
because	causal	why
since	temporal causal	when why
when	causal + temporal temporal conditional	when
although	contrast concession	yes/ no
as a result	result	why
for example	instantiation	give an example where
for instance	instantiation	give an instance where

Figure 3: Discourse Connectives [2]

Mannem et al., 2010; Lindberg et al., 2013 parsed articles and formulated rules on parse trees to generate question. Discourse connectives provides cue word for question generation tasks by determining the question type. [2] Parsed questions have very high mechanical paraphrasing of the context. Such questions lack depth in comprehension. It's mere play on words and their ordering.

Pronoun Resolution 13.54 percent of the generated questions using hand-crafted rules have unresolved pronouns. [2] Below is an example:

Context Tsunamis grow in height when they reach shallower water, in a wave shoaling process.

Answer they reach shallower water

Question Generated When do they grow in height?

Parsing Complexity Complex sentence with three or more connectives or decorators would cause the parser to generate questions with syntax or semantic errors. 9.38 percent of the sentences from the experimented dataset have such problem. [2]

Context In a family who know that both parents are carriers of CF , either because they already have a CF child or as a result of carrier testing , PND allows the conversion of a probable risk of the disease affecting an unborn child to nearer a certainty that it will or will not be affected.

Answer either because they already have a CF child or as a result of carrier testing

Question Generated Why do in a family who know that both parents are carriers of CF , either or will not be affected ?

2.1.2 Question Generation Template

A more generalized approach than parse tree formulae are question template, experimented by Lindberg et al., 2013; Chali and Golestanirad, 2016; Labutov et al. ,2015, which works better on high-level questions. Template based approach

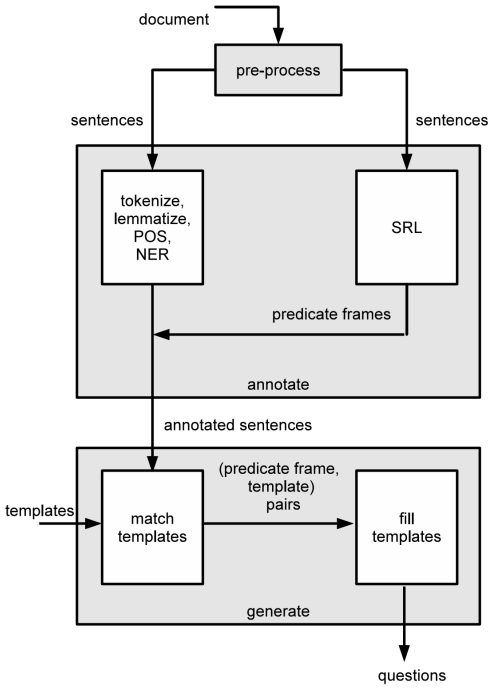

Figure 4: Question Generation Template Architecture [25]

has strong bias. Lindbert elaborated 52 templates. Among those 52 templates,

(21.5%) `Summarize the influence of [A1 -lpp !comma !nv] on the environment.`

(10.9%) `How can [A0 -lpp !nv] [V !be !have lemma] [A1]? ## [A2 null]`

(10.8%) `What can [A0 -lpp !nv] [V !be !have lemma]? ## [A1] [A2 null]`

(8%) `What [V !be !have pres tps] [A1 !nv]? ## [A2 null] [A0]`

(6.8%) `What [V be] [A1 -lpp !nv !comma]?`

Figure 5: Top 5 Question Templates

five contributed 58 percent for all the question generated while six templates

had never generated any questions. In fact, the most used template was not even in a question form, as shown in Figure 4. [25]

2.2 Activation Function

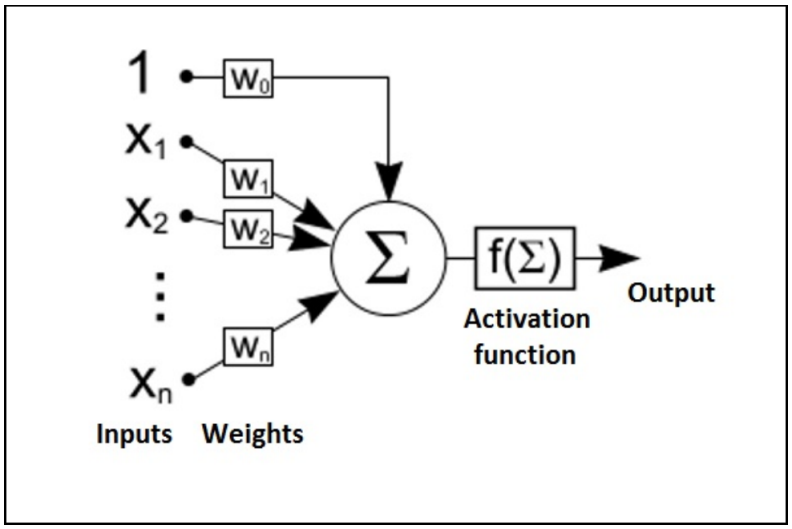

Figure 6: A Typical Artificial Neuron: an artificial neuron contains the weighted sum of inputs, fed into an activation function for outputting to add non-linearity.

Activation function (as shown in Figure 6) transforms the weighted inputs with their bias values into an output signal. [33] Activation functions are non-linear, differentiable, and monotonic. Like most neural network models [28], the BERT language model uses RELU as its main activation function and sigmoid as the output activation function. A typical neural network model uses a combination of RELU and sigmoid activation functions.

2.2.1 Sigmoid

A sigmoid function is a differentiable monotonic increasing function. [17] Sigmoid functions are often the activation functions in the output layer due to their smoothness in computing real-valued labels. Figure 7 shows most frequently used sigmoid functions.

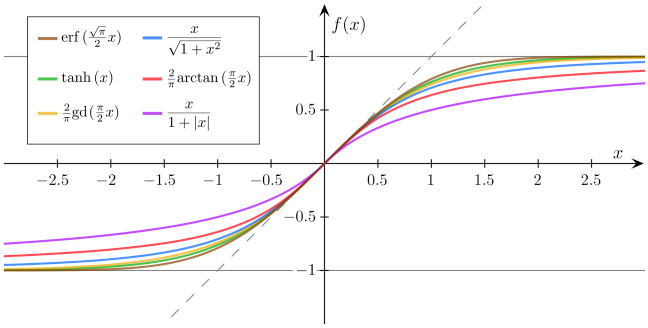

Figure 7: Common Sigmoid Functions: sigmoid functions get their names due to their "S" shapes

$$\sigma(x) = \begin{cases} max(0, x) & , x >= 0 \\ 0 & , x < 0 \end{cases}$$

Figure 8: RELU: the rectified linear unit is y=x with negative values clipped to 0.

2.2.2 RELU

RELU stands for rectified linear unit. It is a simple linear function. [6] RELU is the most used activation function for the hidden layers in the neural network model. [1] The sigmoid based model has the vanishing gradient problem. Unbounded activation functions, such as RELU and softplus, could reduce the gradient decay during the back-propagation of a neural network. [34] Neural network models with unbound activation functions are universal estimators, which means that they can estimate any continuous function. [34]

2.3 Neural Networks

This subsection gives an overview of fundamental neural network architectures. Gradient based learning is a machine learning technique to construct a decision surface to classify high-dimensional patterns. [23] Neural network with back-propagation is a popular example of gradient based learning.

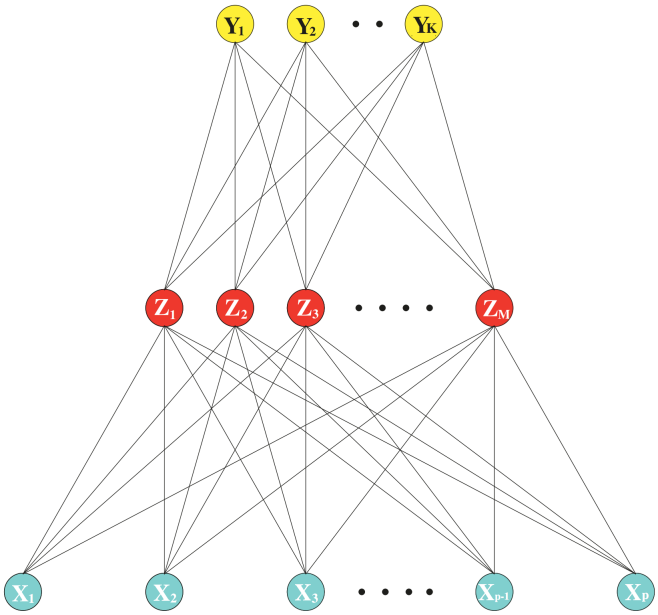

Figure 9: Single Layer Perceptron

2.3.1 Single Layer Perceptron

Single layer perceptron is the vanilla neural network, which was first described by Warren McCulloch, a neurophysiologist. As shown in Figure 9, Xs are the layer of inputs. Zs are the layer of hidden states. Ys are the layer of predicted values. Multi-Layer Perceptron would have more than one layer of hidden states $Z = A(W * X + B)$. A stands for activation function, which gives nonlinearity to the linear combination of input X and its weight matrix W added with bias matrix B. Figure 9 highlights the computation graph for a single neuron. [18]

2.3.2 Convolution Neural Network

This section explains the basics of convolution neural network, which is used in Section 2.4.2. The main ideas behind convolution neural network are local connections, shared weights, pooling and multi-layers. Convolution neural network has filter banks and feature maps to capture the hidden connections in the data inputs. Feature maps are the outputs of filters in the CNN. Filter banks refer to the sets of weights used in the convolution process, which discovers the local

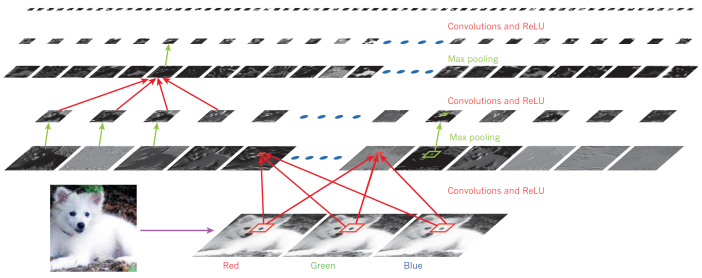

Figure 10: Visualization of Convolution Neural Network Process: a Samoyed dog image has three color channel red, green, and blue. Each rectangular image is the feature map with a layer feature bank. The first layer identifies low level implicit features. The second layer (a pooling layer) reduces the dimension and merges similar low level features into condensed features. The model keeps repeating the convolution and the pooling process until it outputs the task label in the last sigmoid prediction layer.

connections within the data. Pooling chooses a value, such as max value or average value, within a window in the feature maps. The pooling layers merge semantically similar features into one feature. [15] Multi-layer neural network enables the model to learn an hierarchy of abstractions from the input data automatically, without requiring an human expert to hand engineer such rules. The ConvNets take inspiration from the visual neural science. The stacking of convolution and max pooling follows LGN-V1-V2-V4-IT in the ventral pathway. [15]

2.3.3 Recurrent Neural Network

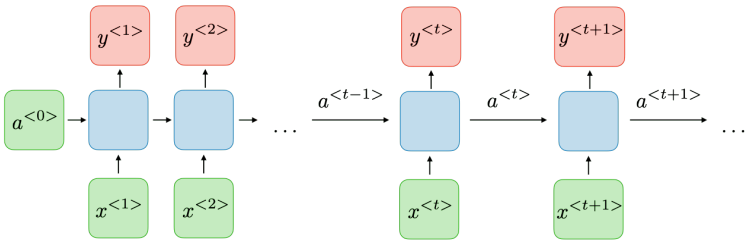

Figure 11: depicts a typical RNN Architecture

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)$$

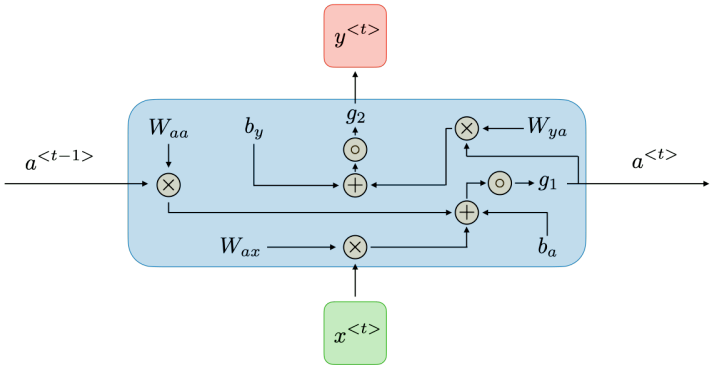

Figure 12: RNN cell

$$y^{<t>} = g_2(W_{ya}a^{<t>} + b_y)$$

Recurrent neural networks are neural network that connect to previous input representations. The goal of RNN is to learn a representation of the input to form a generalized classifier on the task. [4] For each RNN unit (colored in BLEU), input X and previous hidden information flow A are weighted with W and scaled up with a bias B, after which passed to a controller gate G. (Refer to RNN cell Figure 12) The output of controller determines the output target Y. Current RNN unit then passes its hidden state annotation A to next time stamp t.

2.3.4 Long Short Term Memory

Recurrent neural network loses too much information during each time step for long-term dependencies. The training of RNN also takes too much time due to the vanishing gradient problem, whereby a gradient change is too small to cause a significant change to the network, which leads to a premature training stall. To address those issues, LSTM truncates trivial gradients for more than 1000 time steps over a conventional RNN. Cell states C have minor linear transformation from input X at time step t (as shown in Figure 13).

2.3.5 Attention Mechanism

Attention is a vector mapping of the query and the input key and value pairs to a weighted value. It typically takes a key, value, and query. However, for

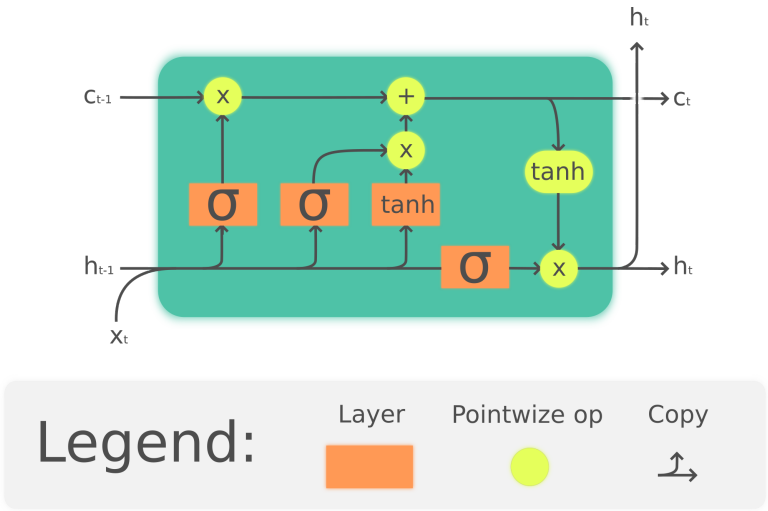

Figure 13: LSTM Cell: the input X of timestep t goes through a four artificial neuron layers (shown as three σ and a tanh) with hidden states from previous timestep t-1.

natural language processing, key and value are often the same input content. Without attention mechanism, an RNN has to encode all the input information into a real-value fixed length vector. By deploying the attention, a decoder of a sequence-to-sequence model can selectively retrieve information from a spread of annotation sequences. [3]

Multiplicative

$$Attention(Q, K, V) = softmax(\frac{Q \odot K^T}{\sqrt{d^K}}) \odot V$$

Q is the query matrix; K is the key matrix; V is the value matrix from the attention input. Dot-product attention may have different normalization scale. The above uses the square root of key dimensions as the scale to normalize the compatibility/alignment score. Comparing to additive attention below, dot-product attention scales up in magnitude as the key dimensions, causing extremely small gradients. Hence, scale deals with the issue by scale small values to larger ones. [37]

Additive

$$\alpha_{ij} = (\frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})})$$

with

$$e_{ij} = a(s_{i-1}, h_j)$$

Additive attention is a shallow feed-forward network. Comparing to dot-product attention, additive attention shares similar attentional property but costs more computation power in implementation, as matrix multiplication is highly optimized. [37]

2.4 Question Generation RNN models

Recurrent neural networks are the building blocks of the first generation neural question generation models. To understand neural question generation, we need to overview RNN, attention mechanism, and LSTM (as done in Section 2.3). The following sections are explanations on the Figure 14.

2.4.1 Embeddings

Word embeddings , also known as word vectors or distributed word representations, are encodings such that the similar words would have similar positions in the vector space. [13]

2.4.2 Knowledge Base

Serban et al. (2016) creates a knowledge base processing system to generate questions from the context with a designated answer using neural network. To apply question generation in database, the head and the relation of a table form the context. The tail in the table functions as the answer. [31] The paper Generating Factoid Questions with Recurrent Neural Networks uses the Freebase as a fact provider for question generation. [31] Freebase is a collaboratively created graph database, with 125,000,000 tuples of 4000 types and 7000 properties, for structuring human knowledge. An http API is available for public read and write for the Freebase graph database. [7]

2.4.3 Question Generation from Images

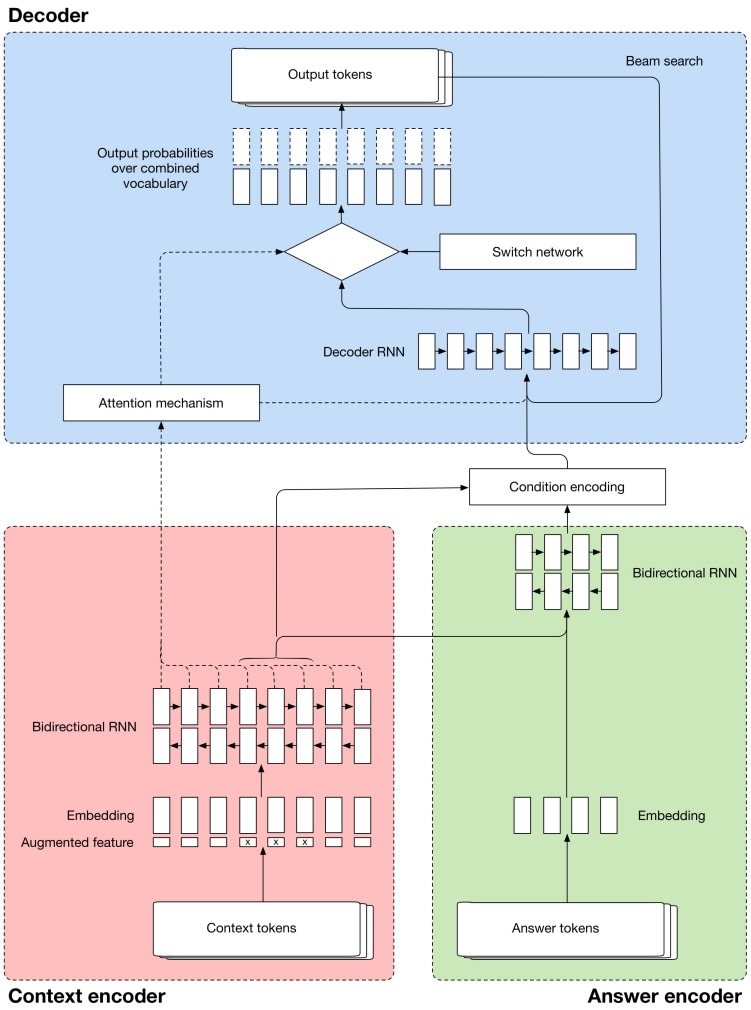

Figure 14: Neural Question Generation Model: NQG models uses a typical seq-to-seq architecture as explained in Section 2.4.4. The input embedding encodes linguistic symbolic into a real-valued vector. Through a sequence of RNN transformation, the NQG model translates the context passage and the answer tokens into a condition encoding. The condition encoding provides question generation information to ask a relevant question. During the question generation process, each token focuses on specific part of the condition encoding through an attention mechanism. To better dealing with out-of-vocabulary tokens, the decoder uses a switcher network to copy words from the input tokens.

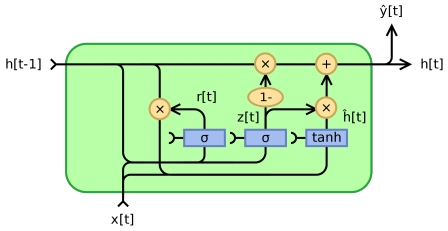

Figure 15: Gated Recurrent Unit Cell

Gated Recurrent Neural Network Figure 15 shows a gated RNN which adds a controller for managing the signal flow among layers. [10] Input X has t time steps. For each time step, the hidden state(h) from previous time step t-1 passes to the current Gated Recurrent Unit. When h(t-1) combines with x(t), the controller decides what to update in its cell hidden state h(t).

Image as Input Mostafazadeh et al. (2016) worked on an image input, instead of text input for question generation, with three datasets in event-centric or object-centric type. They used a Gated RNN, which based on models for image captioning tasks by Devlin et al., 2015 and Vinyals et al., 2015. Their model lacked connection between common sense and image. The pioneering RNN based model for question generation is a collaboration among engineers from Microsoft Maluuba and researchers from Montreal Institute for Learning Algorithms.[42] To train the question generation model, they combined supervised learning and reinforcement learning to model upon SQuAD dataset. For supervised learning, log maximum likelihood function is the optimization target function. For reinforcement learning, it is a policy gradient. The policy gradient contains a performance measurement of a question answering task. This is the very first seq-to-seq based neural network model approach for question generation.

2.4.4 Encoder-Decoder Seq-2-Seq Architecture

"Seq2Seq," is based on the encoder-decoder architecture with attention and pointer-softmax outlined in Bahdanau et al. (2015) and Gulcehre et al. (2016) with separated vocabulary for encoding and decoding. A separated vocabulary means that the encoder has a different vocabulary from the decoder, which causes the output tokens to have more misalignments than models that have

the same vocabulary for both the encoder and the decoder. The Seq2Seq model uses average hidden states from document encoding in the answer positions. Question generation borrows the ideas of encoder-decoder model from machine translation. [3] [9] It's an end-to-end architecture that converts input into a real-valued vector for representation and then uses an decoder for those vectors to make prediction.

2.4.5 BiDirectional Attention Flow Model

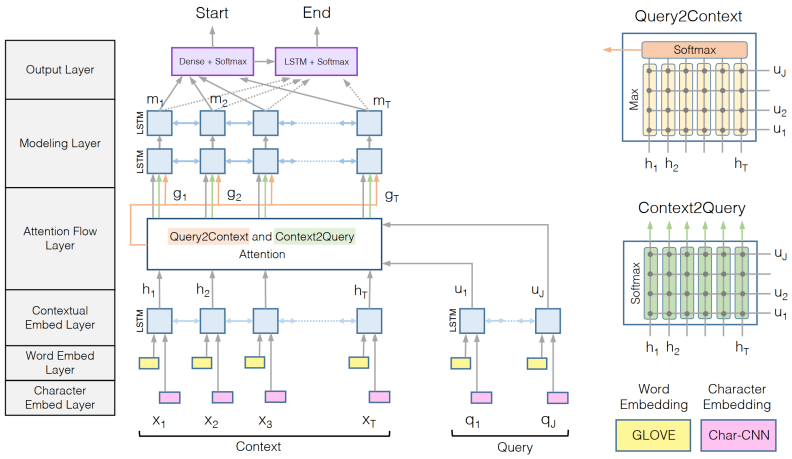

Figure 16: Bi-Directional Attention Flow Model

BiDAF is a multistage context representation model. There are three degrees of contextual encoding: character-level, word-level, and paragraph-level. Neural attention mechanism enables machine learning model to focus on some area of the context to perform deep and targeted processing. BiDAF uses a bi-directional neural attention to encode different levels of contextual information. Bi-direction means that the model computes the attention score both from query to context and from context to query. Instead of storing those information in a fixed vector, BiDAF computes the attention scores for each time step and passes them into deeper layers of the model, avoiding loss in information during the early summarization in stateful attention computations. [30]

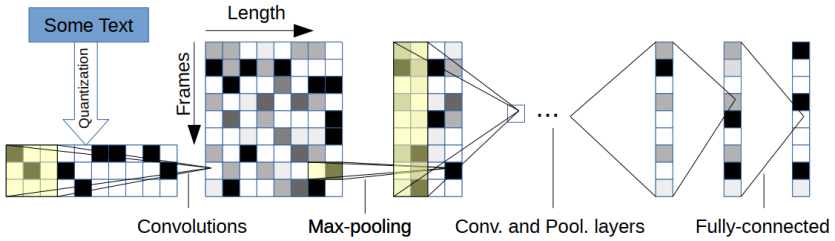

Figure 17: Character Temporal Convolution Neural Network model: this model consists of three parts: the input embedding layer, the context embedding layer, and the output layer. The character embedding is a kind of input layer that maps each word to a vector space with character level CNN. Char-CNN uses a one dimensional temporal convolution neural network to convolute through the text inputs. The word embedding does a similar transformation but with word-level convolutions. The context embedding layer convolutes information into a context space where high dimension coordinates identify particular contextual knowledge for the task. In this CNN based model, the contextual embedding layers are consecutive convolutions and max-pooling filters.The output layer is a vanilla fully connected network, which identifies which token to output [43].

2.4.6 Encoder

The encoder for neural question generation model is based on document and answer embedding vectors. Document(context passage) and answer embedding vectors are sequences of tokens, created by NLTK tokenizer in preprocessing. [3] After the tokenization process, a binary feature vector indicates if the document word belong to the answer. The encoding process for the neural question generation model is similar to the BiDirectional Attention Flow Model. [30]

Annotation Vector Annotation vector is the hidden states produced by bidirectional LSTM network by concatenating forward and backward hidden states. To encode the answer, Neural QG used annotation vector corresponding to the answer word positions in the document.

Extractive Conditon Encoding Extractive condition encoding is the result of concatenating answer word embeddings with corresponding annotation vector, applied to a bidirectional LSTM. The initial encoding states for decoder network is based on annotation vector and extractive condition encoding.

2.4.7 Decoder

The question tokens are generated from a conditional distribution based on previously generated tokens. To refer to the entities in the context, the decoder network deployed a copy mechanism, the pointer softmax (detailed in Figure 18). Pointer softmax consists of two layers: shortlist softmax and location softmax. In beam size K, the decoder searches for question tokens to maximize the result token. The decoder aligns information according the attention mechanism and decodes with beam search algorithm.

2.4.8 Beam Search

Best-first searching algorithm such as A* has large memory consumption or slow runtime. [14] Beam search resolves such issues by keeping sized K branches(beam width) in the search tree using the breath-first search strategy. The memory consumption thus scales linearly with the beam width. With a beam size of a number larger than the amount of branches of a level in the search tree, beam search becomes breath first search. [14] The HARPY system is the first system to use beam search algorithm for addressing the backtracking issue and the redundant computation issue. [29] HARPY researchers describe beam search as a best-few search strategy since it searches only a few "best" paths in parallel without backtracking. [26] For the year 1971, the Department of Defense assigned DARPA a single mission to track Soviet nuclear submarines in open ocean. Beam search was the fastest algorithm to satisfy this goal.

2.4.9 Pointer Softmax

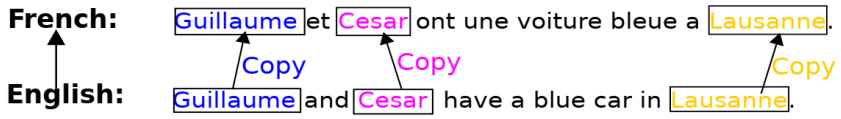

Figure 18: Pointer softmax example: translation task from English to French. The pointer softmax makes prediction to copy source input to output directly.

This section explains the pointer softmax used in the NQG model during the decoding process. Pointer softmax is an attention mechanism to deal with rare or unknown words for generation tasks using neural networks. Predicting

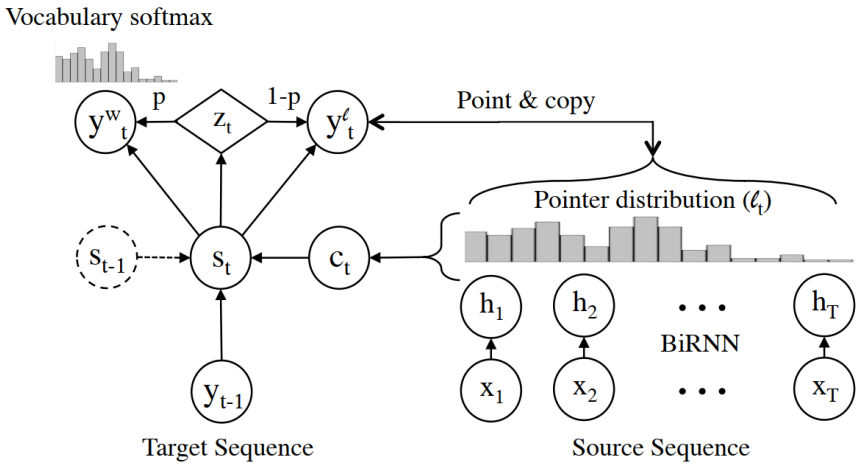

Figure 19: Pointer softmax architecture: Z: switcher network Y: predicted token from vocabulary shortlist (y^w) or copy pointer (y^l)

whether a token uses pointer softmax and indicating the location within the context tokens are the two abilities of pointer softmax. It creates a target token by directly copying from the context or the input. To fight against out of vocabulary prediction, this is an effective method. Pointer softmax is based on psychological evidence of human interaction with unknown objects. It has three components: [16]

Shortlist Softmax Shortlist softmax makes a prediction in a token from the vocabulary shortlist (y^w in 19). This is similar to typical softmax output layer found in generation tasks. [16]

Location Softmax Location softmax is a pointer network which copies token from input text directly, without lookups in the vocabulary shortlist. (y^l in 19) Its output dimension depends on the length of context sequence. [16]

Source Swithching Network The source switching network chooses which decoder to use through training. It is a MLP network. MLP stands for multi-layer perceptron, the vanilla neural network. The inputs for the source switching network are the annotation vector (Section 2.4.6) and previous tokens' hidden states from the RNN decoder. The output is a binary value Z at time stamp t,

where as 1 indicates the using of shortlist softmax or the typical output softmax and 0 indicates the using of location softmax to copy directly from the context tokens. [16] For tokens yet still unknown to both the context and the shortlist vocabulary, the switcher network deploys shortlist softmax to output token 'UNK'. [18]

2.4.10 Results

The models tested in this paper all have low BLEU score, which could be caused by the nature that there are multiple semantically independent questions can be asked on a given context and answer pair.[42]

2.5 Unified Language Model

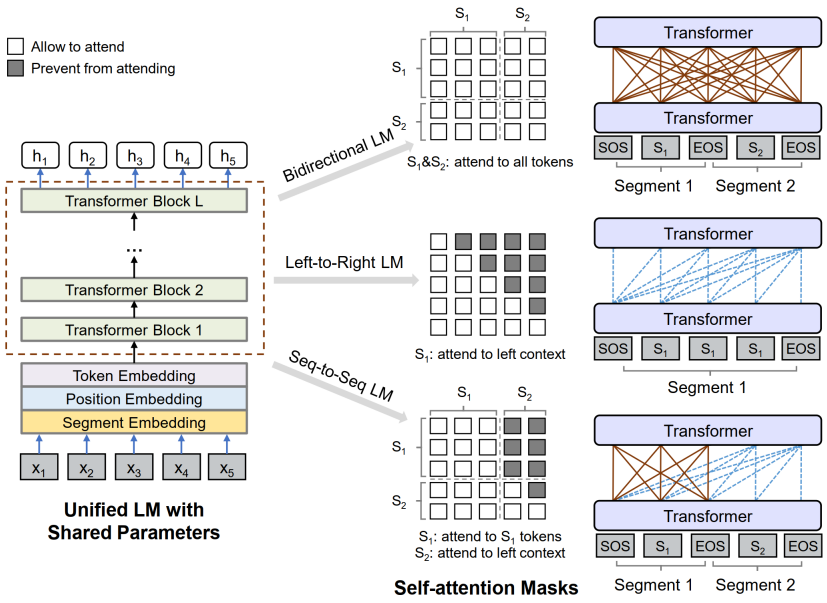

Figure 20: Unified Language Model: ULM unifies multiple downstream language processing tasks into a common upstream model. The upstream model provides the context encoding or annotation vector as hidden states(h).

This section introduces the building blocks of the ULM, including the transformer mechanism, which is a key component for the state-of-the-art models, and the pre-trained language model.

2.5.1 Self Attention

Self-attention is the attention that uses the input as both queries (Q) and keys (K). Instead of finding the relation between a encoder representation and a decoding token, self-attention is capturing the relationship among a given sequence.

2.5.2 Transformer

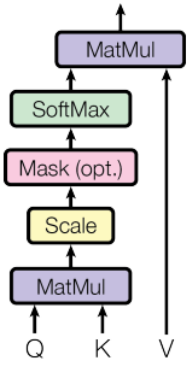

Figure 21: Scaled Dot Product Attention: the scaled dot product attention takes in a query (Q), a key (K), and the value of the input (V). The attention score is a product between a softmax activation score and the input value.

Transformer (shown in Figure 23) stacks multiple attention(shown in Figure 21 and Figure 22) to work in parallel to compose a comparison score. Transformer is an RNN alternative. Its encoder uses self-attentions and feed-forward networks; its decoder uses self-attentions, attention over the encoders, and feed-forward networks. The self-attention replaces the "memory cell" architecture in RNN. The alignment attention remains the same during the generation decoding process. Self-attention works more efficient than recurrent controlling gate. By utilizing attention mechanism only, transformer reduces the model complexity since it removes "memory" hidden unit such as required in GRU. [37]

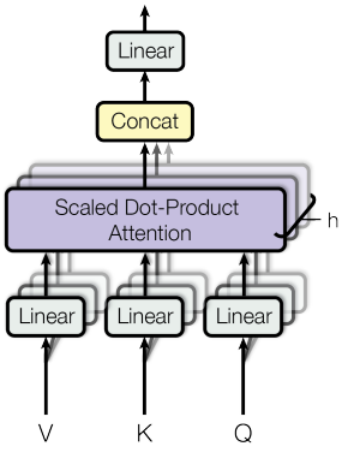

Figure 22: Multi-Head Attention: multi-head attention is a concatenation of multiple (size h) scaled dot product attention.

2.5.3 Language Model

Statistical language model tries to learn the joint probability function for the word sequences. [5] The task for a language model is to predict the next token based on the context inputs. The inputs for language model are task-dependent, typically involving two or more types of sentences. In the case of question generation task, the input has two or three types (details in Section 3.1).

ULM Unified language model pre-trained on left-to-right, right-to-left, and sequence-to-sequence language modeling tasks. A shared Transformer network, along with self-attention masks, unified such tasks by controlling the context exposed. Dong e.t.c. splat the SQuAD 1.1 dataset into training and test sets while keeping the original development set and conducted experiments that used the reverse dev-test split. They used a sequence to sequence model with input passage and answer span as encoder inputs, and with the generated question as decoder output and fine-tuned UniLM on the training set for ten epochs with 32 as batch size, 0.7 masking probability, 2e-5 learning rate, and 0.1 label-smoothing.

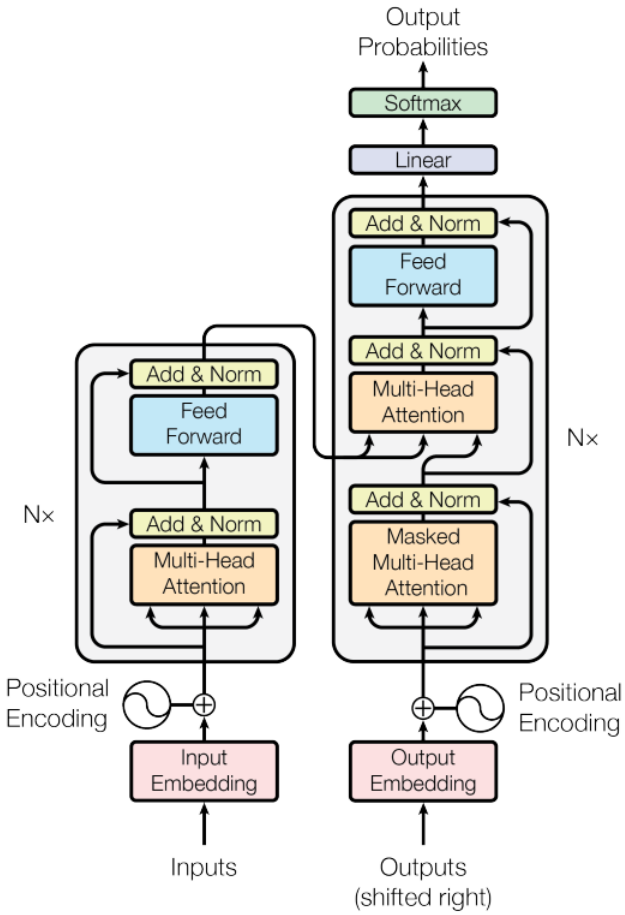

Figure 23: Transformer: the transformer replaces "memory cells" in RNN with self-attentions. Positional encodings provide location information that are lacking in this pure attention setup.

2.6 BERT

BERT stands for bidirectional encoder representation from transformers, which converts a piece of text into real-valued vector space coordinates. It is a pre-trained model on next-sentence-prediction task. [11]

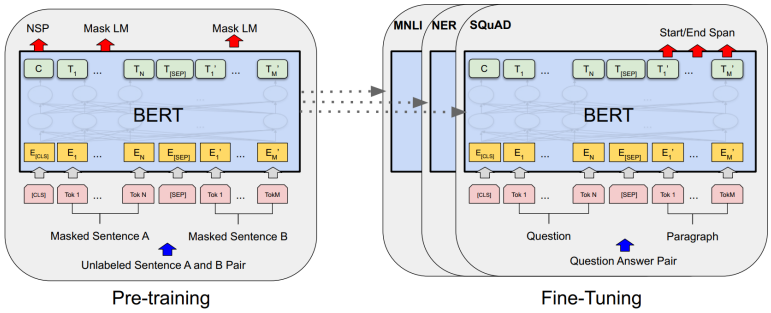

Figure 24: BERT Procedures: The BERT model is a upstream model head. By using different task-specific fine-tuning, the BERT model can solve different kinds of problems, such as classification, neural entity recognition, and the Stanford Question and Answering task.

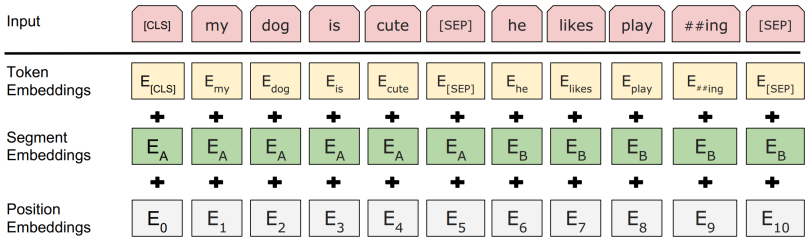

Figure 25: Input Embeddings

2.6.1 Inputs for BERT Model

The BERT model learns information from a real-valued vector space. The input tokens first go through a tokenization preprocessing to transform into an integer indexed dictionary (the shortlist or the vocabulary). After the tokenization process, three types of embeddings feed the position information, the sentence type, and the token embeddings into the BERT model.

WordPiece tokenization breaks the words in the input sentences into a limited set of sub-word segments. [39] With a limited set of word components, the model handles unknown or unlisted vocabulary better than the models with the word-wise shortlist. [39]

Consider the following example:

Input Sentence Jet makers feud over seat width with big orders at stake.

WordPiece Tokens _J et _makers _fe ud _over _seat _width _with _big _orders _at _stake

In WordPiece tokenization, _ indicates the beginning of a word. The spaces are dividers of separate tokens. The training set for the WordPiece model is a training corpus paired with the desired token dictionary D. WordPiece model is a language model optimized to output minimal amount of tokens necessary to segment the whole corpus. [39] The result of the WordPiece model is typically a vocabulary with size between 8k and 32k. As shown in Figure 24, the NSP pre-training accepts a sequence of WordPiece tokens, starting with the reserved classification token ([CLS]). A separator token ([SEP]) divides the two tokenized sentences (A and B). The BERT model would output a real valued vector (C) of hidden size (H) as the classification prediction. [11]

2.6.2 BERT Model Pretraining

The next sentence prediction is a binary classification task where target labels are IsNext or NotNext. The dataset generator picks sentences from a monolingual corpus with half of the data as consecutive sentences. The WordPiece tokenizer then encodes these sentences into WordPiece tokens. The NSP pre-training has conditioned information from both directions (the left to right direction and the right to left direction) to find dependencies within the inputs.

2.7 Recurrent HL-BERT

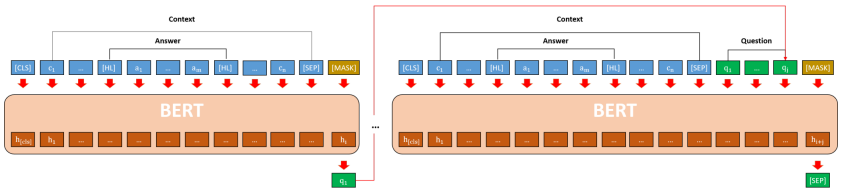

Figure 26: HLSQG: the sequential question model with highlighter tokens generates question tokens as a language modeling task.

Fine-tuning the BERT with a prediction head creates a task-specific output. In the case of the question generation task, the HLSQG paper uses a pre-trained BERT model to generate questions. Chan etc. experimented on three models for question generation task: BERT-QG, BERT-SQG, BERT-HLSQG. The BERT-QG model encodes the context and outputs all the question tokens directly. The Sequential Question Generation model outputs one token at a time by conditioning on previous token, instead. The Highlighter Sequential Question Generation model introduces new highlighter([HL]) tokens to indicate the answer location in the context. The Recurrent HLSQG model yields convincing results, achieving 0.22 in the BLEU-4 benchmarking.[8]

$$X = cls, C, sep, q_1, q_2, ..., q_n, mask$$

Input X is a combination of classification token (cls), context tokens (C), a separator(sep), and a mask token, where the prediction is. Question tokens(q) are the previously generated token recurrently fed into the BERT model.

$$P(q|X) = argmax(softmax(h_{mask} * W + B))$$

Output P is a predicted label index by applying a softmax over the whole vocabulary shortlist on a fully connected predictor layer.

2.8 ProphetNet

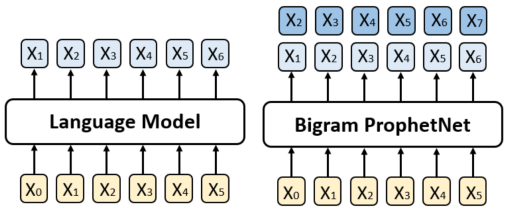

Figure 27: ProphetNet with Bi-gram: the pre-training of ProphetNet processes n-tokens at a time, instead of one token at a time, as found in most language model.

ProphetNet introduced a new objective for language model pre-training: future n-gram prediction. Different from typical language model, instead of

predicting the next token, ProphetNet fine-tunes the model to predict future n tokens, hence avoiding over-fitting over just one token. ProphetNet used a 16GB base dataset and a 160GB large dataset. [41]

3 Cue Word Recurrent Question Generation

The question generation models mentioned above have limited control over generated question types. One way for such a model to direct the generated question type is through the answer given to the context. This can be ineffective when the answer has importance over several types of questions.

3.1 Input

Inputs are sentence piece token encoding for context, sequence type embedding, attention mask vector, output index vector, position embedding.

3.1.1 Token Encoding

The sequence input for the question generation model is a vector of 512 integer values. Each integer represents their word piece mapping in the WordPiece vocabulary file. 0s are padding tokens, ignored by the attention heads in the transformer using attention masks. Suppose we have the following context:

> The black currawong (Strepera fuliginosa), also known locally as the black jay, is a large passerine bird endemic to Tasmania and the nearby islands within the Bass Strait. One of three currawong species in the genus Strepera, it is closely related to the butcherbirds and Australian magpie within the family Artamidae. It is a large crow like bird, around 50 cm (20 in) long on average, with yellow irises, a heavy bill, and black plumage with white wing patches.

If we provide black jay as the answer, the output from the BERT HLSQG model is:"What is strepera fuliginosa also known as?" However, one may have a question asking the color of strepera fuliginosa. It is unclear how to fine-tune the model to "guess" what kind of question to ask before the generation task. To address this issue, we deploy cue word as the starting token during the question generation phase.

3.2 Encoder

The encoder for cue-word question generation is the same as the HLSQG model (detailed in Section 26) BERT encodes the 512 WordPiece tokens into a real-valued vector space through the stacked transformers (detailed in Figure 23). For each token in input sequence, a vector, which defaults to 4096, represents 4096 dimensions of token properties. The representation vector functions as the Extractive Condition Encoding, found in the Neural Question Generation paper. [42]

3.3 Decoder

The masked language model covers some of the tokens from the input. The model then predicts the missing token ids based on their context. The decoder is a masked language model head which outputs prediction distribution for each vocabulary token as a multi-class task. Only tokens marked with output positive index are predicted. An argument max or softmax is then applied to map predictions over the 30,000 token labels back to its corresponding sentence piece token encoding. Recurrently applying the decoding process with beam search, the BERT-based model generates question tokens.

3.4 Cue Word

This section defines the cue word and defines how to use cue words to guide question generation. A cue word is the starting token of the sentence. It can be the word "how" or one of the wh-word such as "what", "who", "why", "when", "where" or beginning of close questions like "is", "are", "does", or "do". Such words have impact on the type of question generated.

3.5 The Model

Cue word BERT model bases on the HLSQG model from Chan etc. in figure 26. Inputs are word-piece token encoding for context, sequence type embedding, attention mask vector, output index vector, position embedding. Token input encoding is a vector of 512 integer values. Each integer represents WordPiece mapping in the vocabulary file. The BERT model encodes input information into a real-valued vector space through transformers. For each token in the

input sequence, a vector, which defaults to 4096, represents 4096 dimensions of token properties. The decoder is a masked language model head which outputs prediction distribution for each vocabulary token as a multi-class task. Only tokens marked with output positive index are predicted. An argument max or softmax is then applied to map predictions over the 30,000 token labels back to its corresponding WordPiece token encoding. Recurrently applying the decoding process with beam search, the BERT-based model generates question tokens starting with the cue word.

3.6 Natural Question Fine-Tuning

3.6.1 Natural Question Dataset

Natural Question dataset consists of Google search queries from real world usage. With each sample having a question statement, a short answer annotation, and a long answer annotation, there are 300,000 training samples in the dataset. A python script parses the NQ dataset to context-question-answer triplets. Based on the HLSQG training script, the training process skips any context that are longer than 512 tokens [20]

3.6.2 Fine-Tuning

A python script parsed the NQ dataset to context-question-answer triplets. Based on the HLSQG training script, training skips context that are longer than 512 tokens.

4 Evaluation

To evaluate the quality of the generated question, the NLG-Eval script reports eight measures: BLEU score up to 4 grams, METEOR, ROUGE, CIDEr, skip thought cosine similarity, embedding average cosine similarity, vector extreme cosine similarity, and greedy matching score. [32] Among those metrics, BLEU-4 is the commonly reported measurement for question generation.

4.1 BLEU Score

$$BLEU = BP * gmmPrecision$$

$$BrevityPenalty = \begin{cases} 1 & \text{if c (candidate length) is longer} \\ e^{(1-r/c)} & \text{if r (reference length) is longer} \end{cases}$$

$$GeometricMeanModifiedPrecision = \sum_{n=1}^{N} w_n * log(p_n)$$

$$ModifiedPrecision(p_n) = \frac{Count_{clipped}(ngram)}{Count(ngram)}$$

Human evaluation is time-consuming and nonreusable. The goal for BLEU is to provide a quick, inexpensive, and language-independent substitute for human effort in evaluating language generation tasks. BLEU score measures how well does the candidate sentence match with the reference sentence in the sequence length, the word of choice, and the word order. BLEU consists of a positive weight, a brevity penalty factor (B.P.) and a modified precision score(gmmPrecision). Brevity penalty factor indicates that the reference sentence (human labeled) has shorter length than the candidate sentence (generated). Precision is the uni-gram match between the generated candidate string and the target reference string. But it over-counts repeated matches. To deal with this issue, modified precision clips off the repeated matches. [27] N-gram refers to the combinations of subsequences for a given sequence. For example:

> The quick brown fox jumps over the lazy dog.

The unigrams for the above sentence are:

> the, quick, brown, fox, jumps, over, the, lazy, dog,.

The bigrams are:

> the quick, quick brown, brown fox, fox jumps, jumps over, over the, the lazy, lazy dog, dog .

The trigrams are:

> the quick brown, quick brown fox, brown fox jumps, fox jumps over, jumps over the, over the lazy, the lazy dog, lazy dog .

The n-gram similarity aims to generalize the longest common subsequence to size of n between two strings. [19] The BLEU-4 score uses 4-gram subsequences from the generated candidate strings and the reference strings to compute.

4.2 METEOR

$$score = (1 - Pen) * F_{mean}$$

$$Pen = \gamma * (ch/m)^{\beta}$$

$$F_{mean} = \frac{P * R}{\alpha * P + (1 - \alpha) * R}$$

$$P = m/t$$

$$R = m/r$$

The METEOR score is a composition of word order penalty (Pen) and parameterized harmonic mean of precision and recall (F_{mean}).

The word order penalty is a parameterized (γ and β) fraction of fragmentation(ch/m). γ decides the maximum of the penalty ($0 < \gamma < 1$) while β (typically 3.0) determines the functional relation.

The segmentation fraction is the number of chunks (ch) of the comparison sequences over the total matches of the words (m). Each chunk has continuous matched word token and identical word order, a.k.a. matching n-grams.

The parameterized harmonic mean is a composition of the precision (P, the percentage of mapped words in the generated sentence) and the recall(R, the percentage of mapped words in the reference sentence), parameterized by α (typically 0.9).

m is the amount of mapped words in the two sentences. A mapping is a word alignment between the two sentences such that a word in a sentence maps to at most a word in the other sentence. There are three mapping calculation modules in METEOR, applied in order: the "exact", the "porter stem", and the "WN synonymy". Exact Module maps words that are exactly the same; Porter Stem Module maps words that have the same stem according to the Porter stemmer. WN Synonymy Module maps words that are synonym according to the WordNet. The optimal mapping has least word order "crossings".

t is the amount of unigrams or words in the translated (or generated) sentence.

r is the amount of words in the reference sentence. [21]

4.3 ROUGE

ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. It measures the overlapping units (n-gram, word sequences, and word pairs) of the machine generated summary to its human reference. There are four variations: ROUGE-N, ROUGE-L, ROUGE-W,and ROUGE-S. [24]

4.4 CIDEr

The design goal for CIDEr is providing a reliable metric for the automatic image description generators. The CIDEr metric system has three components: a triplet-based method to collect human consensus, an automated metric that captures consensi, and two datasets (the PASCAL-50S and the ABSTRACT-50S). [38]

4.5 Cosine Similarities

The cosine similarity between two vectors A and B is their angle in the vector space. The angle between two vectors has an inverse cosine relationship on the ratio between the product of the two vectors and their norms. In this evaluation, we formulate different kinds of similarities from using different kinds of vectors.

4.5.1 Embedding Average Cosine Similarity

Embedding average cosine similarity computes the average of each word embeddings. [32]

4.5.2 Vector Extreme Cosine Similarity

In cars, televisions,and phones with dialog systems, text classification is a key component since it enables such systems to identify the user's intent. [13] Word embedding extrema refers to the coordinates with large absolute values. In the intent recognition task, the domain specific words carry more weights than the daily vocabulary. Those rare words have their word embeddings coordinates further away from the common words. Hence taking word embedding extrema

would stress domain-specific words, which are meaningful in capturing the intents from user dialogs. [13]

$$extrema(d_i) = \begin{cases} max\, d_i & \text{if } max\, d_i \geq |min\, d_i| \\ min\, d_i & \text{otherwise} \end{cases}$$

To get the extrema vector, one takes the maximum or minimum of each dimension d_i in the the word embedding vector dimension set D.

4.6 Greedy Matching Score

The greedy matching of two sentence embeddings calculates the average of the largest possible similarities between each word in the candidate string and the reference string. [32]

$$G(C, R) = \frac{\sum w \in C\, max\, v \in R\, cos\, sim(emb_c, emb_r)}{|C|}$$

$$GM(C, R) = \frac{G(C, R) + G(R, C)}{2}$$

For each word in the candidate string(C), we calculate the maximum possible cosine similarity by a eager match among each word embeddings in the reference string(R). After the summing up those similarity scores, we divide it by the length of the candidate string($|C|$). Flipping the candidate string and the reference string and averaging again yield the greedy match score for those two string embeddings.

4.7 Result

Dataset	BLEU-4	METEOR	ROUGE	CIDEr
81K Paragraph	0.22	0.24	0.49	2.11
Natural Question	0.08	0.13	0.27	0.68

Table 1: BERT Question Generation Result

BERT pre-trained model with HLSQG has a strong baseline, achieving 0.22 in BLEU-4 score. When fine-tuned on natural question dataset, the BLEU score drops dramatically to 0.08, which might be caused by the wider question token

choices, similar to the reasons from the neural question generation paper. [42] With the above example on black jay, the 81k based-model outputs: "what is strepera fuliginosa also known as?" While fined tuned on the natural question dataset, the model generates: "what is the name of the black currawong ". Since the natural question dataset lacks question mark in its question, it might be one cause to the low score after the fine-tuning. What's the impact of question mark? When given a cue word "how", nq-fined-tuned model outputs: "how do you call a bird a currawong" while the original 81k-SQuAD model outputs: "how is strepera fuliginosa known locally ? " The question quality decreases as the BLEU-4 reflected.

4.8 Model Comparison

Model	Question
Rule-Based Model	When do they grow in height?
RNN-Based Model	Why do tsunamis grow in height?
BERT-Based SQuAD	When do tsunamis grow in height?
BERT-Based N.Q.	When do tsunamis grow in depth?

Table 2: Pronoun Resolution Question Comparison(Tsunami Example)

With the tsunami example shown in Section 2.1, the Table 2 shows the questions generated using different models on pronoun resolution issue, which hand-crafted rule-based model has. All neural network based model successfully resolves the pronoun. However, recurrent neural network based model asks a question of the wrong type ("why" instead of "when"). SQuAD fined-tuned BERT model performs the best out of those four models, generating the most natural question. Interestingly, natural question fine-tuned model changes the height to the depth, which is syntactically equivalent. The Bert-based question generation model resolves the pronoun resolution issue better than the rule-based system in Section 2.1. For the Tsunami example, the BERT-based question generation model successfully generates a well-composed question: "when do tsunamis grow in height?", resolving the pronoun "they".

As shown in Table 3, the neural network model outperforms hand-crafted ruled-based model in handling complex sentence example discussed in Section 2.1. The ruled-based model generates: "why do in a family who know that both parents are carriers of CF ,either or will not be affected?". In this example,

Model	Question
Rule-Based Model	Why do in a family who know that...
RNN-Based Model	Why are parents carriers of cf?
BERT-Based SQuAD	When does pnd allow for the conversion of...
BERT-Based N.Q.	When do you know if your parents are carriers of cf?

Table 3: Parsing Complexity Question Comparison(Carrier of CF Example)

the BERT-based question generation fine-tuned on Natural Question dataset performs the best by generating a concise question. The SQuAD fine-tuned BERT-based model asks the most complex question out of these four models: "when does pnd allow for the conversion of a probable risk of the disease affecting an unborn child to nearer a certainty that it will or will not be affected?" The RNN-based model also asks a relevant question but with minor semantic error (parents, instead of ones' parent) in the above example.

4.9 Fine-Tuning Comparison

Consider the following context:

> Westwood One will carry the game throughout North America, with Kevin Harlan as play-by-play announcer, Boomer Esiason and Dan Fouts as color analysts, and James Lofton and Mark Malone as sideline reporters. Jim Gray will anchor the pre-game and halftime coverage.

With answer:

> Kevin Harlan

Both the SQuAD and the NQ fine-tuned BERT-based models show inaccurate answers comparing the reference answer. SQuAD fine-tuned model cannot decode the reference of "the game" while Natural Question fine-tuned model successfully decoded "the game" as NFL football. However Natural Question fine-tuned model uses the "voice" as the announcer, which is less accurate than the SQuAD fine-tuned model.

Source	Question
Reference	Who was the announcer for Westwood One's Super Bowl 50 coverage?
SQuAD	Who is the radio announcer for the game?
SQuAD	Who is the team announcer for the game?
SQuAD	Who is the announcer for the game?
Natural Question	Who is the voice of the NFL football game?
Natural Question	Who is the voice for the 2018 NFL playoffs ?
Natural Question	Who is the announcer for the 2018 NFL Championship ?

Table 4: Evaluation Questions from Different Fine-Tuning

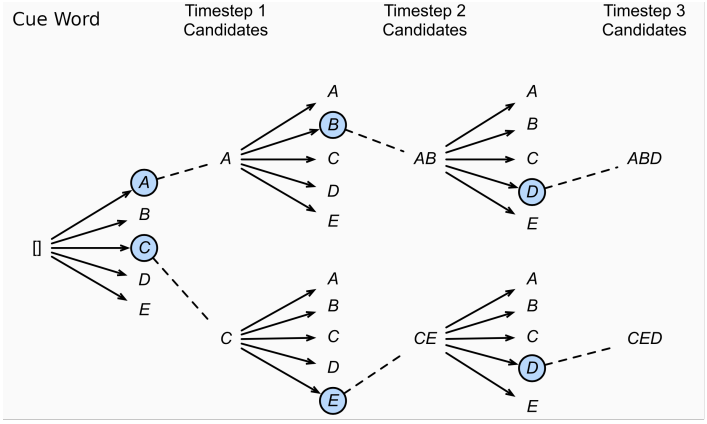

Figure 28: Type Specific Generation: in step Cue Word, the user provides the cue word(A, B, C, D, E). In timestep 1, the beam search algorithm looks for potential candidate tokens (A, B, C, D, E). The search ends when the algorithm finds the token reserved (token "3" in WordPiece encoding) for termination.

4.10 Type Specific Generation

To generate questions of a specific type, the user provides a question type-token of choice (as in Figure 28 and in Section 3.4) as the starting token for the beam search algorithm. Beam search algorithm ranks the candidate question tokens by their losses. Consider the following example:

Context Coronavirus, any virus belonging to the family Coronaviridae. Coronaviruses have enveloped virions (virus particles) that measure approximately 120 nm (1 nm = 10 to the power of -9 metre) in diameter. Club shaped glycoprotein spikes in the envelope give the viruses a crownlike, or coronal, appearance. The nucleocapsid, made up of a protein shell known as a capsid and containing

the viral nucleic acids, is helical or tubular. The coronavirus genome consists of a single strand of positive sense RNA (ribonucleic acid).

Answer Club shaped glycoprotein spikes in the envelope give the viruses a crownlike, or coronal, appearance.

Cue Word	Question
What	What is the characteristic of the coronaviruses?
Why	Why are the viruses in coronaviruses like this?
How	How does the coronavirus look?
Where	Where do the viruses have their heads?
Does	Does the coronavirus have a crownlike or coronal appearance?
Neural Q.G.	What is the name of the club shaped?
SQuAD	What makes up the coronavirus's appearance?
Natural Question	What is the shape of a coronavirus?

Table 5: Cue-Word Question Generation

The BERT-based model takes the context, and the answer provided above on COVID-19 as input and generates question token through beam search. The cue word enables the model to change the questions generated from different types of open questions to closed questions. Within half of a second, the model generates various kinds of questions without complex rule-based parsing (as shown in the discourse connectives section in Figure 3). In the example of Table 5, the Neural Question Generation, the SQuAD fine-tuned model, and the Natural Question fine-tuned model return only "what" questions. The Neural Question Generation model in Section 2.4 generates a question with semantic error (without a subject). The model fine-tuning on the Natural Question dataset appears to make questions generated more brief than the SQuAD fine-tuned model.

5 Future Work

Since fine-tuning BERT model is very hardware dependent, a year is a short time to conduct extensive experiments on this transformer based model. The following would be a decent continuation of the question generation research.

5.1 Question Token Search Strategy

Instead of the typic beam search strategy, other search strategy such as BULB [14] might improve the decoded sentence accuracy.

5.2 Reinforcement Learning on BERT

It would be interesting to implement an reinforcement learning technique on the BERT model to see if it improves.

5.3 High Capacity Model

The latest GPT-3 model has an astonishing 175 billion parameters, which limits the ability for dated hardwares to train such models. It would be interesting to train those models to generate questions.

5.4 Long Context

The training script can handel context up to 512 tokens. For large context, the model has trouble scoping down onto the processing power. A more efficient algorithm on selecting and processing large context would make it possible to experiment on.

5.5 Common Sense Pre-training

As the leaderboard shown in 1, pre-trained language models have great advantages over those models which lack pre-training. A possible solution would be having the "common sense" embedded into the model.

6 Conclusion

Asking non-trivial questions autonomously by artificial intelligence is a solvable task. Cue word gives BERT based sequential question generation model a starting token. It reduces the search space for questions in the output sequence to best fit the need of reading assistance and adds control over the question tokens generated. In this year-long master research thesis, we examined the rule-based approach, RNN question generation model, and Transformer based model such as BERT, ALBERT, RoBERTa, and T5. We also looked at generation quality

measurement metrics, such as the BLEU score. We further fine-tuned the model on Google's Natural Question dataset. I implemented a demonstration utilizing Python Django backend and Flutter frontend with a CUDA GPU server. There were attempts to refine the question generation model on reintroducing positional embedding to the BERT model, SentencePiece highlighter tokenization for ALBERT, and highlighter token removal for ALBERT model. Since there is a GPU training bug in the T5 model library, the experiment with T5 was incomplete. Due to the time constraints, these approaches lack promising results but could be an extension of question generation task in the future. Using reinforcement learning to boost question quality would be worthy of its effort. The lacking of "common sense" is a hindrance to the model to generate natural-sounding questions without contextual information. Adding an upper "slow thinker" module would be of great interest to tackle such a problem.

7 Acknowledgement

I want to thank Dr.Smith on provision of this project and Dr.Frank's permission to use the ML servers. This thesis would be impossible without either of them. I also want to thank the authors of HLSQG paper for providing their code. Those code provided a foundation for experimenting on their results and further fine-tuning. The figures credits belong to their original authors under fair educational use. The RNN figure 12 credits to Stanford CS230 lecture`https://github.com/afshinea/stanford-cs-230-deep-learning`. The leaderboard 1 is from paperswithcode.com question generation`https://paperswithcode.com/sota/question-generation-on-squad11`. The neural question generation RNN model 14 is cited from Mr.Hosking's Github repository `https://github.com/bloomsburyai/question-generation` Unified language model figure 20 is cited from Dong's Microsoft research paper[12]. Figure 27 cites from [41].

References

[1] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[2] Manish Agarwal, Rakshit Shah, and Prashanth Mannem. Automatic question generation using discourse cues. In *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 1–9. Association for Computational Linguistics, 2011.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] Jonathan Baxter. Learning internal representations. In *Proceedings of the eighth annual conference on Computational learning theory*, pages 311–320, 1995.

[5] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[6] Anurag Bhardwaj, Wei Di, and Jianing Wei. *Deep Learning Essentials: Your hands-on guide to the fundamentals of deep learning and neural network modeling*. Packt Publishing Ltd, 2018.

[7] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, 2008.

[8] Ying-Hong Chan and Yao-Chung Fan. A recurrent bert-based model for question generation. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 154–162, 2019.

[9] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[10] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *International conference on machine learning*, pages 2067–2075, 2015.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[12] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, pages 13063–13075, 2019.

[13] Gabriel Forgues, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay. Bootstrapping dialog systems with word embeddings. In *Nips, modern machine learning and natural language processing workshop*, volume 2, 2014.

[14] David Furcy and Sven Koenig. Limited discrepancy beam search. In *IJCAI*, 2005.

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[16] Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. Pointing the unknown words. *arXiv preprint arXiv:1603.08148*, 2016.

[17] Jun Han and Claudio Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International Workshop on Artificial Neural Networks*, pages 195–201. Springer, 1995.

[18] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[19] Grzegorz Kondrak. N-gram similarity and distance. In *International symposium on string processing and information retrieval*, pages 115–126. Springer, 2005.

[20] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*, 2019.

[21] Alon Lavie and Abhaya Agarwal. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the second workshop on statistical machine translation*, pages 228–231, 2007.

[22] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[23] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[24] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[25] David Lennart Lindberg. *Automatic question generation from text for self-directed learning*. PhD thesis, Applied Sciences: School of Computing Science, 2013.

[26] Bruce T Lowerre. The harpy speech recognition system. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE, 1976.

[27] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[28] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[29] D Raj Reddy et al. Speech understanding systems: A summary of results of the five-year research effort. department of computer science, 1977.

[30] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

[31] Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. Generating factoid

questions with recurrent neural networks: The 30m factoid question-answer corpus. *arXiv preprint arXiv:1603.06807*, 2016.

[32] Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *CoRR*, abs/1706.09799, 2017.

[33] P Sibi, S Allwyn Jones, and P Siddarth. Analysis of different activation functions using back propagation neural networks. *Journal of theoretical and applied information technology*, 47(3):1264–1268, 2013.

[34] Sho Sonoda and Noboru Murata. Neural network with unbounded activation functions is universal approximator. *Applied and Computational Harmonic Analysis*, 43(2):233–268, 2017.

[35] Duyu Tang, Nan Duan, Tao Qin, Zhao Yan, and Ming Zhou. Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027*, 2017.

[36] David A Van Dyk and Xiao-Li Meng. The art of data augmentation. *Journal of Computational and Graphical Statistics*, 10(1):1–50, 2001.

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

[38] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575, 2015.

[39] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[40] Dongling Xiao, Han Zhang, Yukun Li, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. Ernie-gen: An enhanced multi-flow pre-training and

fine-tuning framework for natural language generation. *arXiv preprint arXiv:2001.11314*, 2020.

[41] Yu Yan, Weizhen Qi, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. *arXiv preprint arXiv:2001.04063*, 2020.

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

[43] Xiang Zhang and Yann LeCun. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*, 2015.

[44] Xin Zhang, An Yang, Sujian Li, and Yizhong Wang. Machine reading comprehension: a literature review. *arXiv preprint arXiv:1907.01686*, 2019.