NEURAL QUESTION GENERATION WITH TRANSFER LEARNING AND UTILIZATION OF EXTERNAL KNOWLEDGE

MARJAN DELPISHEH

A THESIS SUBMITTED TO
THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

Graduate Program in
Electrical Engineering and Computer Science

York University
Toronto, Ontario

September 2020

Abstract

Question generation (QG) aims to automatically generate questions from text documents. Neural question generation (NQG) applies deep neural networks to solve the problem of QG. Deep neural networks have been successfully applied to many real-world problems, including natural language processing (NLP). However, their performance relies heavily on the availability of a large amount of labelled training data where the desired output of the model to be learned given an input is provided. For domains where labelled training data are very limited, NQG models suffers from poor performance. Another problem that NQG encounters is the problem of rare and unknown words, which are the words that occur during both training and inference phases but do not exist in the vocabulary list of the system. In this thesis, solutions to both problems are presented. We first investigate the impact of transfer learning on NQG on a domain where labelled training data are very limited, and explore the effects of transferring knowledge learned from data in a general domain into different layers of the NQG network. To deal with the rare and unseen word problem, we integrate semantic relationships defined in the WordNet lexical database, which is a type of general knowledge external to the training data, into the input representation of the NQG system. We conduct experiments to evaluate the proposed models and demonstrate significant improvements over the state-of-the-art methods across various evaluation metrics.

Acknowledgments

First and foremost, I would like to thank my supervisor Dr. Aijun An. She provided me with guidance and freedom. Dr. An not only created a great research environment for me and all my colleagues, but also taught me how to do research. I am very grateful for her support and for the insightful discussions.

Thank you to my thesis committee members, Dr. Ruth Urner and Dr. Costas Armenakis, for their constructive comments that made this thesis better.

Finally, I would like to express my deepest gratitude to my family, for always supporting me. I would not be where I am today without the support and love from my family.

Table of Contents

List of Tables

List of Figures

Abbreviations

Below is the list of abbreviations used in **Natural Language Processing** for this thesis:

QG	**Q**uestion **G**eneration
NQG	**N**eural **Q**uestion **G**eneration
NLP	**N**atural **L**anguage **P**rocessing
QA	**Q**uestion **A**nswer
OOV	**O**ut-**O**f-**V**ocabulary
UNK	**Unk**nown
NLG	**N**atural **L**anguage **G**eneration
CBOW	**C**ontinuous **B**ag-**O**f-**W**ords
MLM	**M**asked **L**anguage **M**odel
NSP	**N**ext **S**entence **P**rediction
POS	**P**art-**O**f-**S**peech
NER	**N**amed **E**ntity **R**ecognition
BIO	**I**nside–**O**utside–**B**eginning
PS	**P**ointer **S**oftmax
NMT	**N**eural **M**achine **T**ranslation
BLEU	**Bi**lingual **E**valuation **U**nderstudy
METEOR	**M**etric for **E**valuation of **T**ranslation with **E**xplicit **OR**dering
ROUGE	**R**ecall-**O**riented **U**nderstudy for **G**isting **E**valuation
LDA	**L**atent **D**irichlet **A**llocation
Seq2seq	sequence-to-sequence
BP	**B**revity **P**enalty

Below is the list of abbreviations used in **Neural Networks** for this thesis:

DNN	**D**eep **N**eural **N**etwork
LSTM	**L**ong **S**hort-**T**erm **M**emory
RNN	**R**ecurrent **N**eural **N**etwork
GRU	**G**ated **R**ecurrent **U**nit
BRNN	**B**i directional **R**ecurrent **N**eural **N**etwork
CNN	**C**onvolutional **N**eural **N**etwork
SGD	**S**tochastic **G**radient **D**escent

Chapter One

Introduction

The task of Question Generation (QG) aims to generate a set of question-answer (QA) pairs from raw text without the need of human labor. The generated QAs can be useful for training deep neural network models for building e.g., dialog systems. QG can also benefit reading comprehension (Du, Shao, and Cardie, 2017), self-learning, forming questions to estimate students' knowledge and preparing course materials for the students (Heilman and Smith, 2010). Question generation methods can be classified into two categories: (1) the rule-based approach, e.g,. (Mitkov and Ha, 2003; Heilman and Smith, 2010; Mazidi and Nielsen, 2015) (2) the deep learning approach, e.g,. (Du, Shao, and Cardie, 2017; Zhou et al., 2017; Sun et al., 2018). Traditional rule-based approaches use hand-crafted rules to form questions, which requires labor-intensive formation of rules. Also, the rules generated in one domain may not be transferable across domains. Deep learning approaches have been applied on many natural language processing tasks, such as machine translation, text summarization, etc. These tasks inspired the use of neural network-based methods for questions generation as well. For example, Du, Shao, and Cardie (2017) employed

the sequence-to-sequence model on this task. In general, end-to-end neural-based approaches are more capable of generating complex questions than rule-based methods (G. Chen et al., 2018).

1.1 Motivations

One of the most important characteristics of Deep Neural Networks (DNNs) is that their performance relies on the availability of a large amount of data. Sufficient amounts of training data play an important role when developing these models. Neural network approaches for QG are competitive when a large amount of data is provided. For example, neural question generation (NQG) models trained on SQuAD (Rajpurkar et al., 2016), which is a general purpose question and answer (QA) data set containing about 100,000 QA pairs, achieve good performance. However, the performance of NQG systems is far from satisfaction when only a small set of QA pairs is available for training. This is often the case when generating questions for a specific domain (such as *car manuals*) where labeled data (i.e., existing QA pairs) are scarce. A technique for dealing with small data sets is *transfer learning*, where knowledge learned from one domain is transferred to another. More specifically, in transfer learning, models learned for tasks for which a large amount of labelled data are available are used in learning a model for tasks where labelled data are very limited. Transfer learning has been used in computer vision and some NLP tasks such as text classification, but little investigation has been done on its use in neural question generation, especially on the *car manual* domain.

Another problem with Natural Language Processing (NLP) applications is brought by *out-of-vocabulary* words (OOV). Words play an integral part in building NLP systems. They are the input and/or output to these systems. A very large number of words exists in each language, whose vocabulary is also evolving. The common output layer of NLG systems is a *softmax* whose dimension is the number of words in the vocabulary. In order to reduce the computational complexity of these systems, the vocabulary is often shrunk to contain the top-k most frequent words from the training data (Gülçehre et al., 2016). The words that do not appear in this vocabulary are *unknown words*, which are represented by the $\langle UNK \rangle$ token. These words are referred as the *out-of-vocabulary* words, and they have the same vector representation during training and test time. Therefore, this problem have an adverse impact on many Natural Language Generation (NLG) tasks such as Neural Question Generation.

1.2 Contributions

The objective of this work is to address the above open issues for neural question generation (NQG). Below are the contributions of the thesis:

- We investigate the use of transfer learning in question generation for a domain where labeled data is scarce. In particular, we would like to see whether the QG models learned from a large amount of general-purpose QA pairs can help learn QG models in a specific domain (e.g., car manuals) where labelled data are very limited. In addition, we study the impact of partially transferring

parameters from different layers of the source model, e.g., embedding weights and inner layers parameters and demonstrate the performance of the model after being fine-tuned on the target domain. We also explore the impact of choosing more semantically similar instances to the domain of our interest as the training data.

- We propose a new word representation method for the NQG problem when encountering out-of-vocabulary words during the training and testing phases. Instead of utilizing one vector representation for all the unknown words, we incorporate the general knowledge of human beings into a neural network-based approach for question generation to determine the vector representation of an unknown word. In particular, we utilize the synonym relations between words defined in WordNet (Miller et al., 1990) in the embedding layer of the neural network. In other words, the embedding vector for each unknown word is computed based on its synonym representations existing in the embedding table of our system. Embedding table is a simple lookup table in the system that stores the word embeddings for the words existed in the vocabulary. This table maps each word index to its distributed representation. The words embedding vectors can be obtained from this table using their indices.

- We conduct empirical evaluation of our proposed methods on the state-of-the-art NQG models. The experimental results show that our proposed methods improve BLEU, ROUGE and METEOR scores considerably where the available data for training a deep neural model is not sufficient on the specific domain

of car manuals. We demonstrate the effects of using both techniques, namely parameter initialization as an additional step of training our model using a large QA dataset and employing the wordnet synonym relation on top of our NQG model in chapter 4. The generated questions have a better quality in terms of coherence, grammar and their relatedness to the provided answers compare to our baseline models.

1.3 Overview of the thesis

The structure of the thesis is as follows: In Chapter 2, we discuss types of Question Generation followed by methods for evaluating NQG systems. We further provide a brief introduction of deep learning techniques specifically for text generation. In Chapter 3, we review the related works for neural question generation. Chapter 4 is devoted to our domain adaptation technique for NQG. We also propose a method to tackle the problem with out-of-vocabulary words by incorporating relations defined in the WordNet (Miller et al., 1990) lexical database in our NQG system. The list of the abbreviations used in this thesis are provided on pages x and xi.

Chapter Two

Background

This chapter provides background knowledge to the concepts of the subsequent chapters. It reviews techniques used in natural language processing, mostly neural networks, followed by fundamentals of deep learning. It then introduces the foundation of common deep learning architectures in NLP.

2.1 Word Embedding

Word embedding is a technique to represent words as vectors of real numbers. It is used in many natural language processing tasks such as sentiment classification, question answering system, text summarization and question generation.

2.1.1 One-hot Encoding

One-hot representation is an integer vector representation of a word w in the vocabulary V. The dimension of this vector is equal to the size of V, i.e., w is represented in the $R^{|V|\times 1}$ dimensional space. The value of the vector at the corresponding

index for w is 1 while the other elements have the value 0 in the sorted order for the English language. However, in this representation the similarity between the word vectors is not considered. However, it would be useful to have representations that can be used to capture semantic similarities between words.

2.1.2 Embedding matrix

Embedding matrix E maps each word in the vocabulary of the system to a vector representation where $E \in R^{|V| \times d}$, where d is the embedding dimension. The embedding matrix can be learnt through probabilistic a neural language model (Yoshua Bengio et al., 2003) or other techniques such as Word2vec (Mikolov, Sutskever, et al., 2013), GloVe (Pennington, Socher, and C. Manning, 2014), etc. In many NLP applications, the word representations are obtained from the embedding matrix. With these featurized representations we can capture analogies such as *"king is to queen as man is to woman"*. As depicted in Figure 2.1 [1] the concept that discriminates *sir* from *madam* would also distinguish other word pairs such as *man* and *woman*. This discriminative concept could be *gender*.

2.1.3 Language Model

Language models compute the probability of words occurring in a sequence, denoted as $P(w_1, w_2, ..., w_n)$. This technique generates words and phrases that are more likely to appear in the same context together. The probability of a word appearing at a certain location of the sequence depends on the previous words:

[1] https://nlp.stanford.edu/projects/glove/

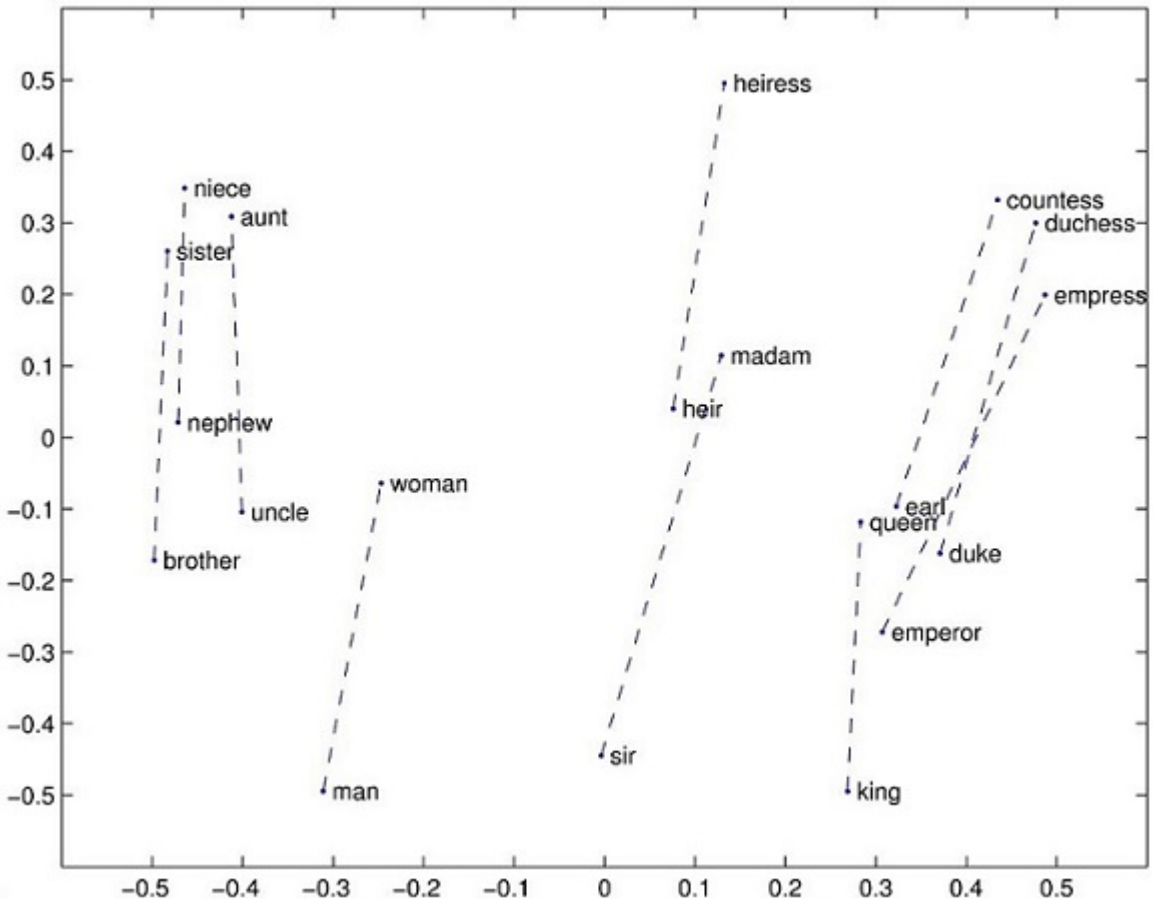

Figure 2.1 GloVe model's word representation by (Pennington, Socher, and C. Manning, 2014)

$$P(w_1, w_2, ..., w_n) = \prod_{i=1}^{n} P(w_i | w_1, ..., w_{i-1}) \tag{2.1}$$

(1) *n*-gram Language Models

The n-gram language model calculates the probability in equation (2.1) by considering the count of the n-grams. More specifically, the probability of a word occurring

at the i-th location from the n-gram language model is computed as follows:

$$P(w_i|w_1,...,w_{i-1}) = P(w_i|w_{i-1},...,w_{i-(n-1)}) \approx \frac{count(w_{i-(n-1)},...,w_{i-1},w_i)}{count(w_{i-(n-1)},...,w_{i-1})} \quad (2.2)$$

where $count(x)$ is the number of occurrences of sequence x in a training corpus of text. There are a few issues with the proposed n-gram models. One is caused by the *Sparsity* problem, which is if the n-gram is not seen before, then the numerator or denominator in equation (2.2) become zero. In order to address this problem, techniques such as *smoothing* (S. F. Chen and Goodman, 1996) and *back-off* (Katz, 1987) are used. Another problem with the n-gram language model is that the counts of n-grams in the corpus need to be stored. This leads to the storage problem.

(2) Neural Probabilistic Language Model

Yoshua Bengio et al. (2003) introduced a neural language model which aims to learn distributed representation of words as well as the probability distribution over the vocabulary given context word embeddings as well:

$$P(w_t|w_{t-n+1},...,w_{t-1}) = \frac{exp(y_{w_t})}{\sum_{w \in V} exp(y_w)}$$
$$y = b + Wx + Utanh(d + Hx)$$
$$x = (C(w_{t-n+1}),...,C(t-2),C(t-1)) \quad (2.3)$$

Let m be the number of features, $C \in R^{|V| \times m}$ is the matrix of distributed word representations that are learnt during training the model. *b, d, W, U* and H are the parameters of the model. The architecture of this model is shown in Figure 2.2.

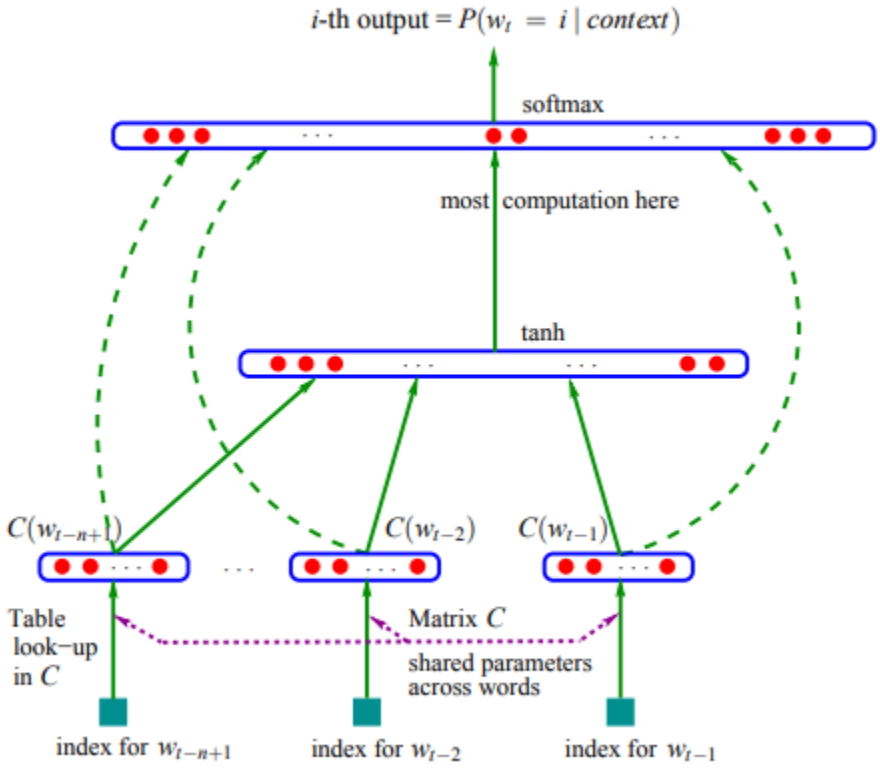

Figure 2.2 The first architecture of deep neural network model for natural language processing introduced by (Yoshua Bengio et al., 2003)

2.1.4 Word2vec

Word2vec (Mikolov, K. Chen, et al., 2013) is another model for learning the distributed representation of words. It utilizes either two model architectures: Continuous Bag-of-Words Model (CBOW) and Continuous Skip-gram Model, Figure 2.3. In the former model, a word is being predicted based on the surrounding context words, while in the skip-gram model the surrounding words are predicted based on the current word. The output layer of this model is softmax which is computationally

expensive due to the normalization over all the vocabulary of the system. Therefore, hierarchical softmax and negative sampling (Mikolov, K. Chen, et al., 2013) could be replaced with the normal softmax layer.

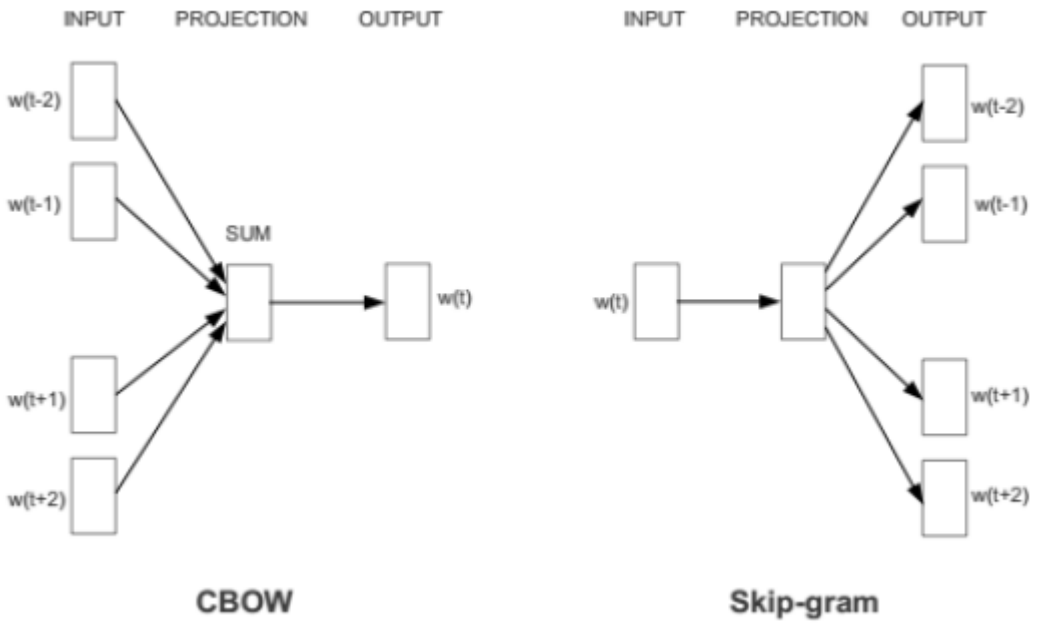

Figure 2.3 CBOW and Skip-gram architectures introduced by (Mikolov, K. Chen, et al., 2013)

2.1.5 GloVe

GloVe (Pennington, Socher, and C. Manning, 2014) is an unsupervised learning algorithm for deriving the distributed word representations. Instead of considering only surrounding context words to generate the word embeddings, this model takes the global word co-occurrence statistics into account.

2.1.6 Contextual Representation

(1) Embeddings from Language Models (ELMo)

Word embeddings derived from pre-trained models such as GloVe or word2vec do not consider the context into account (Peters, Ammar, et al., 2017; McCann et al., 2017). ELMo (Peters, Neumann, et al., 2018) representations learn the word vectors through a bi directional LSTM that is trained with a coupled language model (section 2.1.3) objective on a large text corpus. A forward language model computes the following word in a sequence, while the backward language model predicts the previous token.

(2) Bidirectional Encoder Representations from Transformers (BERT)

Bidirectional Encoder Representations from Transformers (BERT) is a language representation model. The semantic of a word is well understood when the context that the word is placed into is considered. Word embeddings such as Word2vec (section 2.1.4) and GloVe (section 2.1.5) allow downstream tasks to leverage linguistic information which is learnt from a larger corpus. However, one of the limitations to these methods is that they do not take the context of the word into account. ELMo (section 2.1.6) and ULMFiT (Howard and Ruder, 2018) are types of deep contextualized word representations. These context-dependent representations are obtained through learning via bi directional language modeling (section 2.1.3). The problem with these models is that they do not take both previous and subsequent words into account at the same time. In order to make good predictions and capture a deeper semantic of the the context, this task is vital. BERT uses transformer's

encoder (section 2.3) that solves this problem by reading the entire sequence of words at once. BERT is trained using two objectives: masked language model (MLM) and next sentence prediction. **Masked Language Model (MLM):** a percentage of the input tokens are masked at random and they are replaced with $[MASK]$ token and the model predicts masked tokens with cross-entropy loss. **Next Sentence Prediction (NSP):** During the NSP task, the model tries to predict whether two sentence are followed by each other or not using binary classification loss. This pre-trained model can be fine-tuned with an additional output layer for various tasks such as natural language inference and question answering systems.

2.2 Recurrent Neural Network (RNN)

Traditional neural networks do not consider the dependency between words. In other words, the connection between words is not integrated within the representation of a sentence. Recurrent neural networks tackle this problem by incorporating loops within their architecture. Hence, the information can be passed through different time steps and the network is aware of the earlier steps.

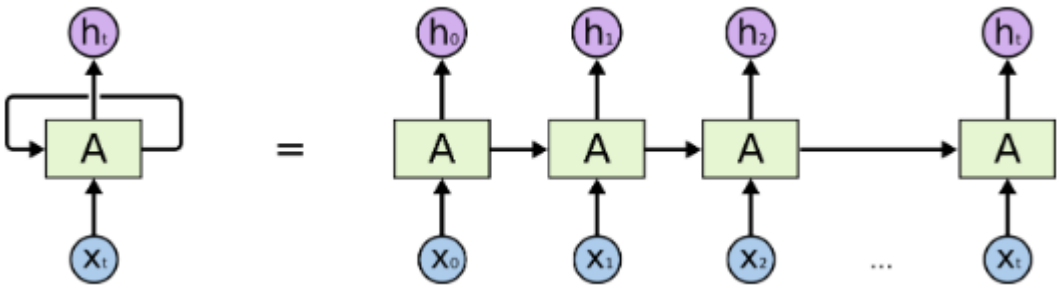

Figure 2.4 Unrolled recurrent neural network

Figure 2.4[2] demonstrates an unrolled recurrent neural network at different time steps. Considering the RNN as function f, it outputs information (hidden state) at time step t, h_t, as follows:

$$h_t = f(x_t, h_{t-1}) \tag{2.4}$$

where x_t is the input word vector such as Word2Vec (Mikolov, Sutskever, et al., 2013), GloVe (Pennington, Socher, and C. Manning, 2014) or a vector embedding learnt from scratch, etc., at time step t. h_{t-1} is the hidden state from the previous step and $h_0 = 0$. To train this network, the loss function at time step t is usually cross entropy:

$$L(\hat{y}^{<t>}, y^{<t>}) = -\sum_{i=1}^{|V|} y_i^{<t>} log\hat{y}_i^{<t>} \tag{2.5}$$

where $\hat{y}^{<t>}$ is the softmax output and $y^{<t>}$ is the target word. The overall loss is the average over all time steps for each individual prediction:

$$L_{total} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{|V|} y_i^{<t>} log\hat{y}_i^{<t>} \tag{2.6}$$

2.2.1 Long Short Term Memory

Learning long-term dependency by RNNs becomes more difficult as the context becomes longer (Y. Bengio, Simard, and Frasconi, 1994). Long short term memory (LSTM) (Hochreiter and Schmidhuber, 1997) is a special kind of RNN suitable for tasks involving long-term dependencies. LSTM contains of four different layers in its architecture instead of a single layer as in RNN (see Figure 2.5 [3]).

[2]https://colah.github.io/posts/2015-08-Understanding-LSTMs

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

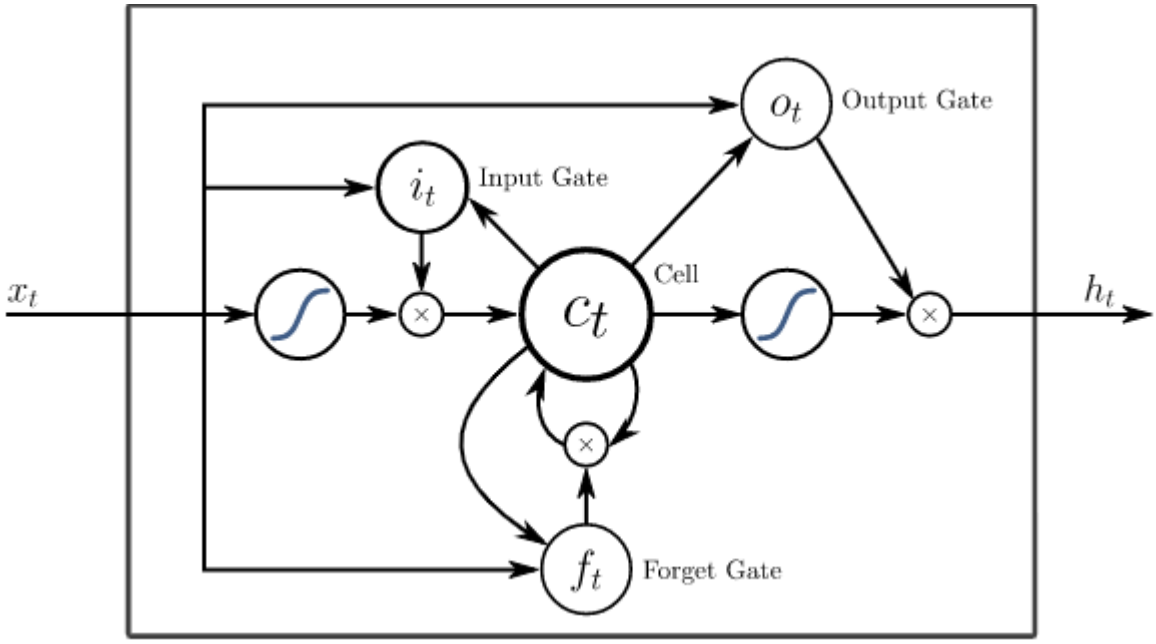

Figure 2.5 Different layers in a LSTM cell

LSTM consist of input gate i, forget gate f, output gate o and cell state C. Assuming g as the LSTM function:

$$h_t = g(x_t, h_{t-1}) \tag{2.7}$$

where h_t computes the hidden state of the current time step and $h_0 = 0$. g is derived by the following computations for different layers (Hochreiter and Schmidhuber, 1997):

- Forget gate layer f_t, controls the information from the previous step that would remain at time step t, equation (2.8). It outputs a value between 0 and 1, where 1 means to keep all the information and 0 means to eliminate the information completely:

$$f_t = \sigma(W_{fx}x_t + W_{fh}h_{t-1} + b_f) \tag{2.8}$$

where W_{fx}, W_{fh} and b_f are learnable parameters and x_t is the word input representation at step t.

- Input gate layer i, computes the values that would be updated. The tanh layer determines the new vector $\tilde{C}_t$ for cell state C.

$$i_t = \sigma(W_{ix}x_t + W_{ih}h_{t-1} + b_i) \tag{2.9}$$

$$\tilde{C}_t = tanh(W_{cx}x_t + W_{ch}h_{t-1} + b_C) \tag{2.10}$$

where W_{ix}, W_{ih}, W_{cx}, W_{ch}, b_i and b_C are learnable parameters.

- The new cell state C_t is calculated by forgetting parts of the previous cell state and updating the new vector value $\tilde{C}_t$:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{2.11}$$

where $\odot$ is the element-wise product. The cell state C_t is passed through a tanh layer so that the values would be within the range -1 and 1. The output prediction h_t is derived by the output gate layer o_t:

$$o_t = \sigma(W_{ox}x_t + W_{oh}h_{t-1} + b_o) \tag{2.12}$$

$$h_t = o_t \odot tanh(C_t) \tag{2.13}$$

where W_{ox}, W_{oh} and b_o are learnable parameters.

2.2.2 Gated Recurrent Unit

Gated recurrent unit (Kyunghyun Cho et al., 2014) (GRU) is a variation of LSTM. Unlike LSTM, GRU has 2 gate layers that are *update* and *reset*, and it does

not contain cell state. GRUs have less complex architecture in contrast to LSTMs and therefore they are computationally less expensive. r_t is the *reset* gate at step t that determines whether the previous hidden state should be ignored or not. z_t is the *update* gate which controls the amount of information transferred to the current hidden state from the previous state.

$$r_t = \sigma(W_{rx}x_t + W_{rh}h_{t-1})$$
$$z_t = \sigma(W_{zx}x_t + W_{zh}h_{t-1}) \tag{2.14}$$

where W_{rx}, W_{rh}, W_{zx} and W_{zh} are learnable parameters. h_t is the new hidden state:

$$\tilde{h}_t = tanh(W_{hx}x_t + W_{hh}(r_t \odot h_{t-1}))$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{2.15}$$

where W_{hx} and W_{hh} are learnable parameters.

2.2.3 Bi directional RNN

In order to expand the information captured from the input sequence (e.g., a sentence) to the network, hidden layers of opposite directions are connected. The forward RNN generates forward hidden states $(\overrightarrow{h_1}, \overrightarrow{h_2}, ..., \overrightarrow{h_I})$ while the backward RNN generates backward hidden states $(\overleftarrow{h_1}, \overleftarrow{h_2}, ..., \overleftarrow{h_I})$. Bi directional recurrent neural network (BRNN) (Schuster and Paliwal, 1997) is trained simultaneously on positive and negative time directions that are forward and backward states respectively (depicted

in Figure 2.6). This feature becomes vital when the context of the input is required to accomplish a task (e.g., dialogue systems, summarization). Therefore, by considering the previous and the following steps, system would have a better understanding of the context. Finally, the hidden state for time step i is $h_i = [\overrightarrow{h_i}; \overleftarrow{h_i}]$. $[x; y]$ represents the concatenation of x and y.

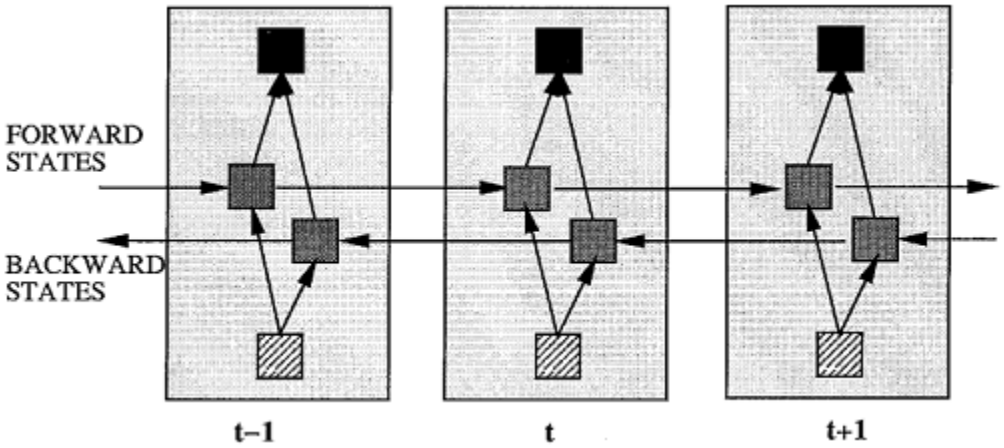

Figure 2.6 Unfolded bi directional RNN at three different time steps (Schuster and Paliwal, 1997).

2.2.4 Encoder-Decoder Architecture

In order to apply RNN to tasks where input and output sequences have various lengths (e.g., summarization, question generation, speech recognition), encoder-decoder structure (Kyunghyun Cho et al., 2014) is proposed. The idea is that one RNN (*encoder*) encodes the sentence into an intermediate representation while another RNN (*decoder*) converts the representation to another sequence.

(1) Encoder

The words in the input sentence is converted to word embeddings. This representation is then passed to the encoder (simple RNN, LSTM or GRU) to generate hidden states:

$$h_i = f(x_i, h_{i-1}) \tag{2.16}$$

where f is a non-linear activation function, x_i is the word embedding at step i. h_0 is initialized to 0. Therefore, the input sentence with length n is converted to intermediate representation $h = \{h_1, h_2, ..., h_I\}$. The encoder RNN outputs whole encoded input representation as the context vector v.

(2) Decoder

The decoder (simple RNN, LSTM or GRU) takes the context vector v and generates decoder's hidden state s_j at each time state of the decoder j:

$$s_j = g(s_{j-1}, y_{j-1}, v) \tag{2.17}$$

where g is a non-linear activation function, s_{j-1} is the previous decoder hidden state and y_{j-1} is the previous generated output. y_0, usually denoted by $< s >$, is the beginning of the decoder state. The goal of the decoder is to estimate the probability of sequence $(y_1, y_2, ..., y_J)$ given the input sequence $(x_1, x_2, ..., x_I)$. The conditional probability of y_j is:

$$p(y_j|v, y_1, y_2, ..., y_{j-1}) = softmax(Ws_j + b) \tag{2.18}$$

where W and b are learnable parameters and $softmax(x_i) = \frac{exp(x_i)}{\sum_j^{|V|} exp(x_j)}$

2.2.5 Attention

The RNN encoder-decoder architecture performs well on short sentences, however as the sequence becomes longer, the performance of the model decreases (KyungHyun Cho et al., 2014). For this purpose, (Bahdanau, Kyunghyun Cho, and Yoshua Bengio, 2014) extend the simple encoder-decoder structure to align and translate jointly for the task of machine translation. At each time step of the decoder, model take the relevant part of the source input as well as the previous hidden state into consideration. More specifically, the model generates a word based on the new context vectors corresponding to input positions (see Figure 2.7):

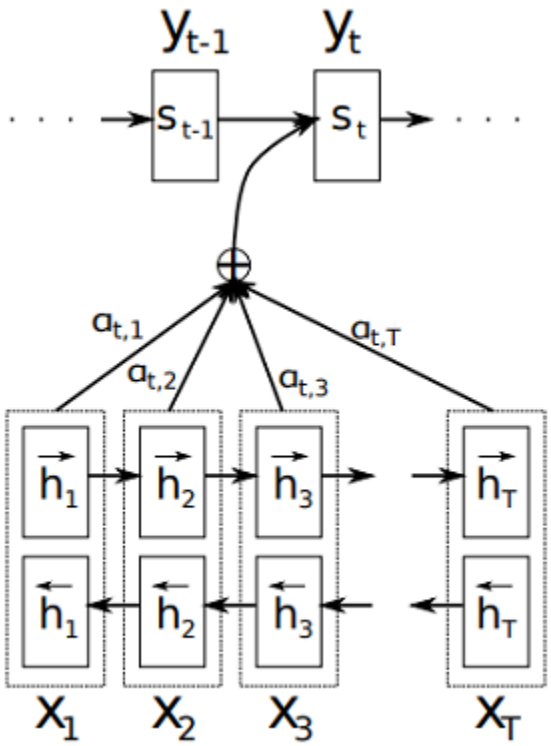

Figure 2.7 Generation of target word y_t given the source sequence $x_1, x_2, ..., x_T$ (Bahdanau, Kyunghyun Cho, and Yoshua Bengio, 2014)

$$v_j = \sum_{i=1}^{I} \alpha_{ij} h_i \tag{2.19}$$

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{i'=1}^{I} exp(e_{i'j})} \tag{2.20}$$

where e_{ij} corresponds to the relevance of i-th step of the encoder and j-th step of the decoder. The relevance is known as *"alignment"* in the context of machine translation. To calculate the score for the relatedness between different time steps of the encoder and the decoder, the scoring functions are as in equation (2.21).

$$score(h_i, s_j) = \begin{cases} h_i^T s_j & \text{dot product} \\ h_i^T W_a s_j & \text{multiplicative attention} \\ W_a^T tanh(W_h h_i, W_s s_j) & \text{additive attention} \end{cases} \tag{2.21}$$

where W_a, W_h and W_s are learnable parameters. Having the scoring function, α_{ij} (i.e., attention distribution) is derived by softmax function over encoder states at j-th step of decoding the output as in equation (2.20). Therefore, we compute the weighted sum of the encoder states as in equation (2.19) to obtain the context vector v_j. v_j can be used during decoding instead of using only the last hidden state of the encoder. Equation (2.17) is therefore modified as:

$$s_j = g(s_{j-1}, y_{j-1}, v_j) \tag{2.22}$$

There are multiple approaches to compute the alignment of inputs at position i and the output at position j as proposed in (T. Luong, Pham, and C. D. Manning, 2015). By having *Global attention*, the model consider all the hidden states of the

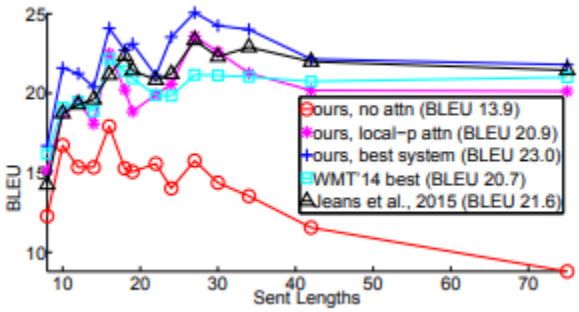

Figure 2.8 The performance of different neural machine translation (NMT) models on long sentences (T. Luong, Pham, and C. D. Manning, 2015)

encoder to compute the context vector. However, this approach is computationally expensive through backpropagation when dealing with large sequences e.g., paragraphs (T. Luong, Pham, and C. D. Manning, 2015). *Local attention* considers a subset of all encoder states to derive the context vector and it is easier for the model to be trained on. Figure 2.9 demonstrates an alignment model that computes the attention weight $\alpha_i j$ between the ith word of the source sentence (English) and the jth word of the target sentence (French).

2.2.6 Coverage Mechanism

One of the common problems during sentence decoding in sequence-to-sequence models is repetition (Mi et al., 2016; Sankaran et al., 2016). See, P. J. Liu, and C. D. Manning (2017) addresses this problem in the abstractive text summarization task by employing the *coverage mechanism* introduced by (Sankaran et al., 2016). The

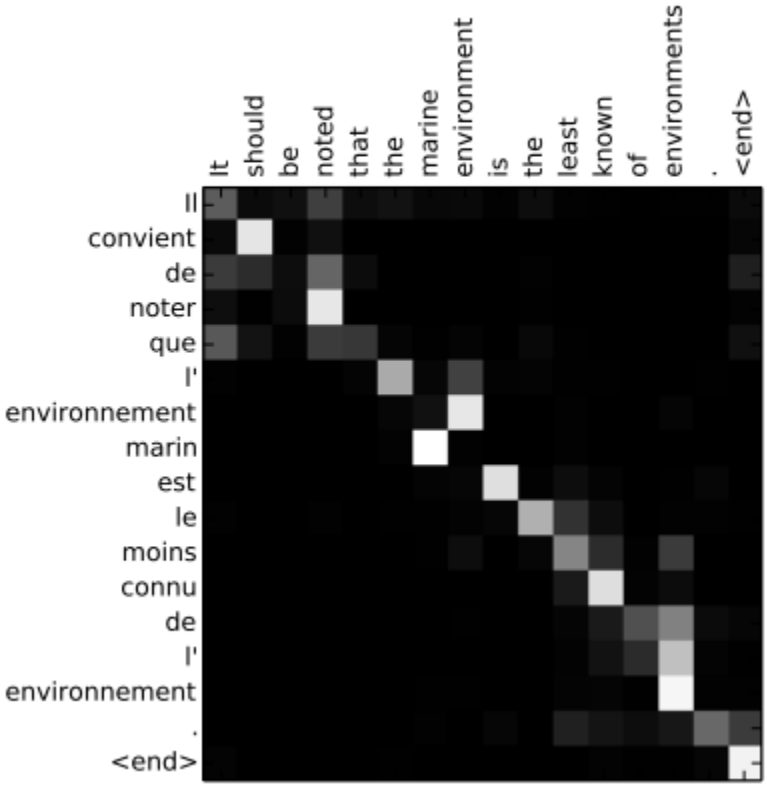

Figure 2.9 Alignment model for neural machine translation (Bahdanau, Kyunghyun Cho, and Yoshua Bengio, 2014)

attention given to each piece of the input is summed up as follows:

$$c_j = \sum_{j'=0}^{j-1} \alpha_{j'} \tag{2.23}$$

c_t is the distribution over the source document that the words have been so far received. c_0 would be initialized to zero. The coverage vector would be considered in computing the attention score (equation 2.21), therefore the next prediction is aware of the words that have been covered during decoding up to step t. The new attention

score is calculated as:

$$score(h_i, s_j) = W^T(W_h h_i + W_s s_j + W_c c_{ij} + b_{attn}) \tag{2.24}$$

Finally, the coverage loss is defined as:

$$covloss_j = \sum_i min(\alpha_{ij}, c_{ij}) \tag{2.25}$$

This term would motivate higher attention weight when the word has not covered so far, since the loss would be still low. In contrast, if the word has been covered during decoding, the attention weight would have a lower value.

2.3 Transformer Models

Transformer (Vaswani et al., 2017) model is originally represented for the sequence problems. This method replaces the RNN based models with the attention mechanism. The architecture of this model consists of two parts: encoder and decoder that are composed of a stack of identical layers, as illustrated in Figure (2.10).

2.3.1 Transformer Encoder

The encoder in the transformer model consists of a stack of $N = 6$ layers. Each of encoder's layer has two sub-layers: multi-head self-attention and a position-wise fully connected feed-forward neural network. A residual connection (He et al., 2015) is employed around each of the sub-layers followed by layer normalization (Ba, J. R. Kiros, and Hinton, 2016). To compute the sequence representation, self-attention

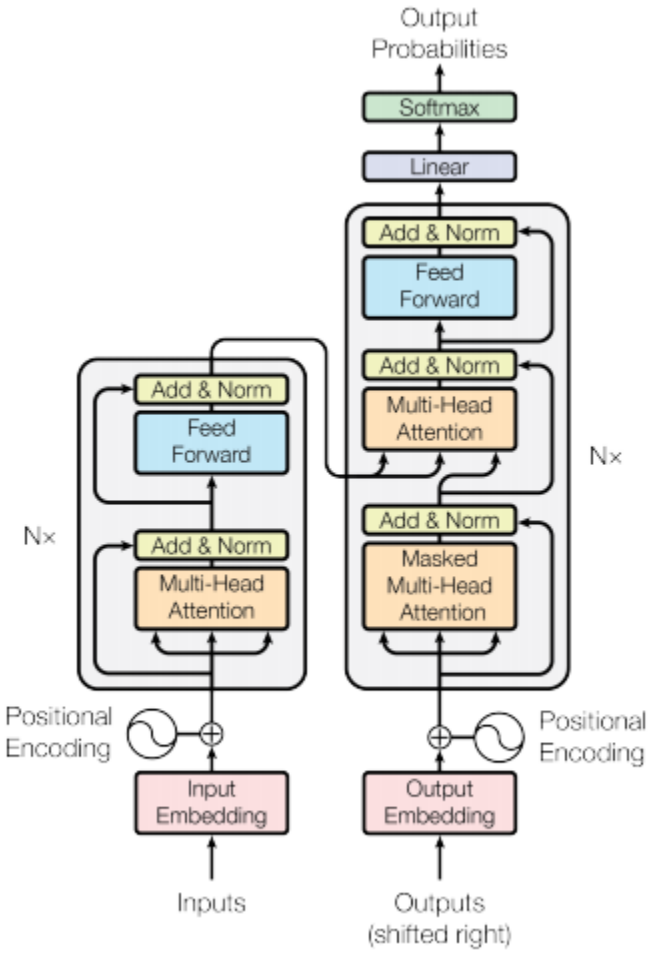

Figure 2.10 Transformer model architecture (Vaswani et al., 2017)

mechanism considers the relevance of different positions of words in a sequence. Self-attention has been incorporated into different NLP tasks such as natural language inference (Parikh et al., 2016), abstractive summarization (Paulus, Xiong, and Socher, 2017). Each token in the sequence $X = \{x_1, ..., x_n\}$ computes attention weights over all the other tokens in the same sequence. All word vectors in the input sequence are projected into *key*, *query* and *value* vectors. These vectors are obtained

by multiplying the embedding vector of each word (x_i) with three learnable matrices W^Q , W^K and W^V. The query and key vectors help to calculate the self-attention scores (i.e,. weight each token in the input) as follows:

$$e_{i,j} = \frac{(x_i W^Q)(x_j W^K)^T}{\sqrt{d_k}} \tag{2.26}$$

where W^Q and W^K are parameter matrices for query and key vectors respectively and d_k is the dimension of the key vector. *i,j* are different locations of the words in a sequence. The result is further passed through a softmax function and the output for the token at location i is computed as follows:

$$o_i = \sum_{j=1}^{n} softmax(e_{i,j})(x_j W^V) \tag{2.27}$$

where W^V is a parameter matrix for the value vector. Since the transformer does not follow the RNN atchitecture, in order to consider the word orders in a sequence, *positional encoding* is encoded into the input embeddings. The position of the words in each sequence are represented as linear combinations of each other.

2.3.2 Transformer Decoder

The decoder is also composed of $N = 6$ layers. Each layer consists of three sub-layers, multi-head self-attention followed by encoder-decoder attention and position-wise fully connected feed-forward neural network. Residual connection and layer normalization are employed around the sub-layers in decoder as well. During the training phase, subsequent words in the sequence are masked. This mechanism

confirms that the prediction at a certain location of the sequence depends only on the previous words. Vaswani et al. (2017) performs multi-head attention, where attention mechanism is computed multiple times with various query, key and value weight matrices (W_i^Q, W_i^K and W_i^V respectively, where i is the ith attention head). The results are further concatenated together.

2.4 Sequence Model Decoders

In a sequence-to-sequence task, the model's objective is to output the best sequence $\hat{Y}^*$ given another sequence, where Y is the target output:

$$\hat{Y}^* = argmax Prob(Y|X) \tag{2.28}$$

The search space to find such sequence can be large. There are a few techniques to shrink the size of the candidates and form the output sequence:

2.4.1 Greedy Decoding

Greedy decoding is an algorithm which chooses the most probable word according to the model's vocabulary. However, this model does not guarantee to select the most probable output throughout decoding. It considers a small search space of the possible words. Moreover, if an incorrect word is selected at a certain location, generating the upcoming words are highly dependent on the wrong choice.

2.4.2 Beam Search

Beam search is another algorithm used for generating the most probable sequence during decoding. This method is commonly used in Neural Machine Translation. At each time step of the decoding, this algorithm chooses the top β most common candidate of words, where β is the hyperparamter of the model. The precision of choosing candidates during decoding is increased if β is chosen to be a large number and leads to a better result. However, this approach is computationally expensive, since a larger number of candidates are estimated at different location of the words.

Chapter Three

Related Work

3.1 Question Generation

People are able to ask creative and rich questions in different domains in their everyday life (Rothe, Lake, and Gureckis, 2017). Question generation (QG) aims to create questions from databases, text documents, etc. QG can benefit the education system by providing questions and answers (QAs) as reading comprehension materials (Heilman and Smith, 2010). It can also assist chatbots to actively ask questions in a conversation. Some question-answering systems use a set of QAs as their knowledge base for interactively answering questions that a user may ask. For example, the knowledge base for the *netpeople Assistant Platform* developed by iNAGO Inc.[1] uses a set of QAs generated from the text documents in a domain as part of its knowledge base to enable interactive question-answering with the user when the user's intent is not well captured by the question that the user asks. Manually creating such a knowledge base from the text documents is labor-intensive and time-consuming.

[1] http://www.inago.com

Tools for automatic questions generation would greatly speed up the creation of such a knowledge base for the interactive QA system.

Automatic question generation from texts is a challenging task. Generating coherent, grammatically-correct questions requires a good level of language understanding. Existing automatic QG methods can be classified into two categories: *rule-based approaches* and *neural network approaches* (also called *neural approaches*).

3.1.1 Rule-based Question Generation

Traditional question generation approaches focused on two fundamental aspect for generating a question: *"what to ask"* and *"how to ask"* (L. Pan et al., 2019). The former requires for machines to highlight important parts of the input text while the latter focuses on generating coherent and grammatically correct questions. These traditional approaches take either semantic (Chali and Hasan, 2015; Yao, Bouma, and Zhang, 2012) or syntactic (M. Liu, Calvo, and Rus, 2010; Heilman, 2011) parsing to generate intermediate representations of the text. These representations are further fed into template-based methods (W. Chen and Aist, 2009; Rokhlenko and Szpektor, 2013) to form questions. The syntax-based question generation methods depend heavily on NLP tools. Wolfe (1976) proposed a syntactic-based approach for automatic question generation to enhance an independent study of textual contents. Gates (2008) utilizes NLP software to parse the document and recognizes named entities with BBN's IdentiFinder (Bikel, Schwartz, and Weischedel, 1999), PropBank (Palmer, Gildea, and Kingsbury, 2005), and so on. This system generates reading comprehension questions using human-defined transformation rules. It is tested on

the CBC4Kids corpus (Leidner et al., 2003) which contains news texts for children. Heilman and Smith (2010) presented a question generation system where the questions are first over-generated and are then further ranked. However, their system relies on hand-crafted rules. These approaches do not consider semantics of the text and cannot be used for different domains of study. The meaning of the sentence is sometimes required for the sentences to be parsed accurately to form rich questions. Chali and Hasan (2015) introduced a topic to question method. The method utilizes named entity and semantic role labeling information and further apply rules to create questions. Mazidi and Nielsen (2014) generate questions and answers from sentences using sematic role labels. These architectures heavily rely on the intermediate transformations and templates to form questions which would be difficult to be applied to data in other domains of study.

3.1.2 Neural Question Generation

Recently, neural approaches for the task of question generation are commonly used to tackle issues brought by traditional QG methods such as their reliance on the hand crafted rules. End-to-end architectures can be optimized to address both *"what to ask"* and *"how to ask"* aspects to form a question. Given the input answer $X = \{x_1 , x_2, ... , x_I\}$ where x_i represents the i-th word in the sentence, the task of an NQG system is to generate target question $\hat{Y} = \{y_1 , y_2, ... , y_J\}$ where y_j is the j-th word in the question. The model focuses to find the best question $\hat{Y}$ such that it maximizes the conditional probability given the answer X:

$$\begin{aligned}\hat{Y} &= \underset{Y}{argmax} Prob(Y|X) \\ &= \underset{Y}{argmax} \sum_{j=1}^{J} P(y_j|X, y_{<j})\end{aligned} \tag{3.1}$$

where words generated in $\hat{Y}$ are either from the input answer X or from the vocabulary V.

Most NQG approaches benefit from the sequence-to-sequence framework (Sutskever, Vinyals, and Le, 2014). They also incorporate the attention mechanism (Bahdanau, Kyunghyun Cho, and Yoshua Bengio, 2014) so that the decoder has access to the entire input source while predicting the output, and the copy mechanism (Gülçehre et al., 2016) to address the problem with unknown words. Learning to generate questions via sequence-to-sequence models is a challenging task. This task requires deep understanding of the context and the target facet to ask about. QG can be applied on knowledge bases (Reddy et al., 2017) or on images (Mostafazadeh et al., 2016). Mostafazadeh et al. (2016) introduced the Visual Question Generation (VQG) task. In the proposed task the system would ask questions about an image. Du, Shao, and Cardie (2017) first presented trainable end-to-end question generation model via attention-based sequence to sequence learning method (Bahdanau, Kyunghyun Cho, and Yoshua Bengio, 2014). The proposed model is motivated by neural approaches on machine translation (Sutskever, Vinyals, and Le, 2014), text summarization (Rush, Chopra, and Weston, 2015), etc. The model is experimented on the SQuAD (Rajpurkar et al., 2016) dataset and does not depend on NLP tools or handcrafted rules generated by humans. The rules are usually associated with

specific domains, and also it is expensive to create such rules. The input sentence is fed into the RNN encoder and the question is generated via the RNN based decoder. The model focuses on specific parts of the input sentence while predicting the question based on the global attention (T. Luong, Pham, and C. D. Manning, 2015). This is similar to how people ask questions in general by taking specific parts of the input sentence into consideration as well as the context that the sentence is placed into. Zhou et al. (2017) further enhanced the sequence to sequence model with position aware encoding and lexical features such as word cases, part-of-speech (POS) tagging and named-entity recognition (NER) using the Stanford CoreNLP v3.7.0 (C. Manning et al., 2014) (see Figure 3.1). Position aware encoding indicates the answer span by the inside–outside–beginning (BIO) tagging scheme, where B specifies the start of an answer, tag I is assigned to words in the answer span and tag O determines words that are not a part of the answer. This information is encoded as an extra feature indicating the positions of the answer in a context. They also incorporate the copy mechanism (Gülçehre et al., 2016) in their model. The copy mechanism produces the probability of copying words from the source input at every time step. Their experiments demonstrate the positive effects of these modifications.

(Duan et al., 2017) presented their work for NQG using convolutional neural network (CNN) and recurrent neural network (RNN). They also leverage the generated questions to enhance existing question answering systems. Sun et al. (2018) improved the performance of the feature-enriched pointer-generator model by introducing a question word distribution which is generated based on the answer position in the paragraph. Moreover, they argued context words closer to the answer are

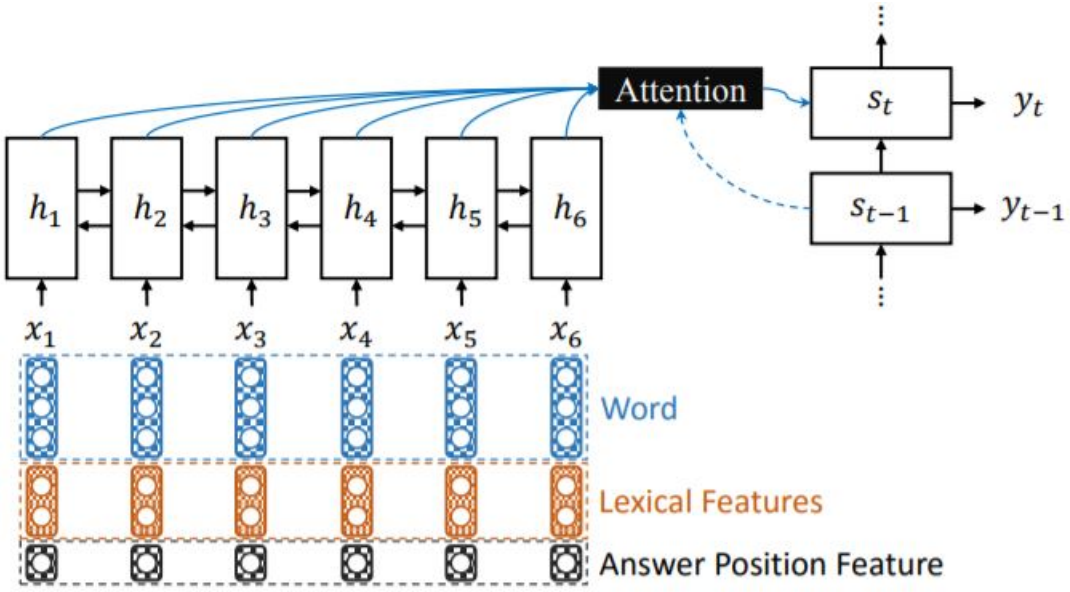

Figure 3.1 Neural Question Generation (NQG) framework proposed by (Zhou et al., 2017)

more relevant and accurate to be copied and therefore deserve more attention. They modified the attention distribution by incorporating trainable positional word embedding of each word in the sentence w.r.t its relative distance to the answer. Figure 3.2 demonstrates the hybrid model by (Sun et al., 2018) where *position-aware attention distribution* aims to copy the words closer to the answer and the *question word distribution* sample question words that are related to the answer and their surrounding words. (Harrison and Walker, 2018) incorporated sentence embedding as well as the linguistic information such as named entity recognition and coreference resolution using (Finkel, Grenager, and C. Manning, 2005; C. Manning et al., 2014). Sentence embedding provides additional information at the sentence level as well as the word level. They demonstrate refinement in generating questions with their

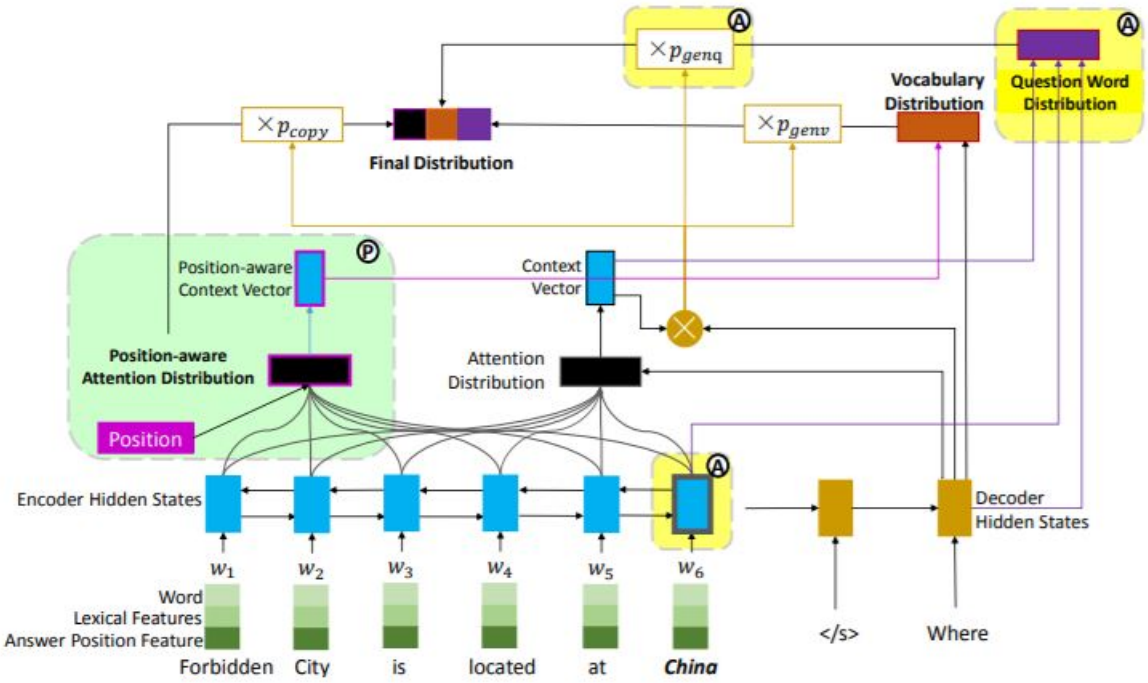

Figure 3.2 Hybrid model for neural question generation proposed by (Sun et al., 2018)

system by conducting human evaluation. Zhao et al. (2018) assist QG by utilizing paragraph-level information with gated self-attention encoder and maxout pointer. A self-matching representation is first derived from the input as in (W. Wang et al., 2017). The maxout pointer mechanism avoid repetition during decoding. Details of this model are discussed in section 4.1. Kim et al. (2018) address the problem of generating unrelated questions to the source context by existing NQG models. They propose the *answer-separated sequence to sequence* by which the passage and the answer are treated individually to improve the provided information. The aforementioned methods are data-driven and bring good performance if the available target

data is sufficient. Given context or sentence, in order to train deep models that performs well on predicting questions, a large amount of available labelled dataset is required.

3.1.3 Dealing With Rare and Unseen Words

Most NLP approaches utilize softmax layer as the output of their network. This softmax layer might compute the probability of each word in the vocabulary list in the output at different time steps. Computing high dimensional softmax is expensive, therefore the vocabulary of the system is usually shrunk to contain top-K most frequent words in the training data. The words that are not in the vocabulary are replaced with the $< unk >$ token during both training and test time. This would impact the performance of the model negatively, and bring issues such as difficulties to understand a good representation of the input data. By containing all the words in the training data into the vocabulary list, we might still observe words that do not appear in the training data. The words that are outside of the vocabulary are replaced with $< unk >$ token. The network might output $< unk >$ tokens during decoding. This is referred to unknown word problem. To address this problem, (Gülçehre et al., 2016) proposed pointer softmax (PS) by which the network is able to copy words from the source sentence. As depicted in Figure 3.3, this model generates a switching variable z_t which decides whether to copy from the source text or to choose the word from the list of the vocabulary. In other words, the switching variable decides to predict the word by copying from the input based on the pointer distribution or to sample from the vocabulary softmax. To this end,

the model utilizes two softmax layers, *shortlist* softmax and *location* softmax. The location softmax points to the location of the source text from which the word would be copied, while the shortlist softmax is the same as the softmax output layer based on which the word will be sampled from the vocabulary.

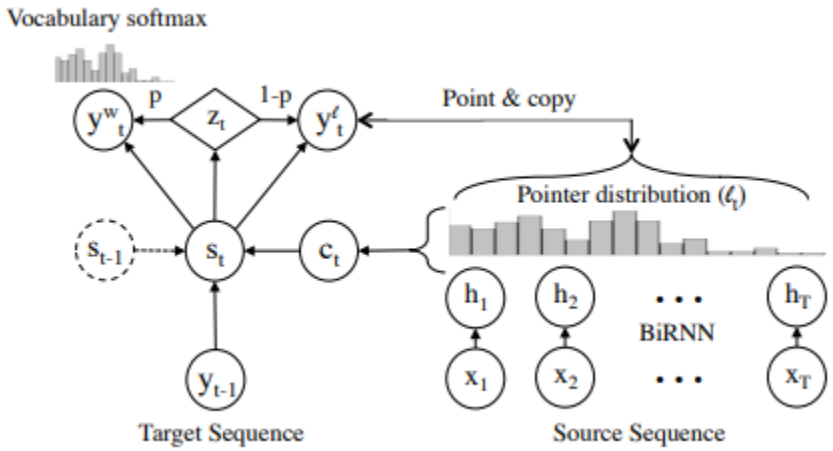

Figure 3.3 Pointer Softmax (PS) architecture by (Gülçehre et al., 2016)

(Sennrich, Haddow, and Birch, 2015) proposed a technique to address the problem with unknown words. The method would represent the unknown words as sequences of subword units by incorporating a compression algorithm that is *Byte Pair Encoding* introduced by (Gage, 1994). Another approach to deal with the rare and unknown words is the work presented by (Ling et al., 2015). Word embeddings are derived by feeding the character level embeddings into bi directional LSTMs. Finally, the final word embedding is obtained from the forward and backward hidden states. M. Luong and C. D. Manning (2016) also deal with the unknown word problem by *Hybrid Word-Character Model*. This model incorporates both word and character level models. It computes the word representation and recover the unknown target

words if it is needed. Figure 3.4 shows an example of the word-character model for the task of NMT (English to French). This model translate the sentence at the word level. However, for the $< unk >$ words in the source sentence a deep LSTM model is learnt over the characters. The last hidden state of the LSTM is the representation of the word. If the model produces $< unk >$ token during decoding, the character level decoder would also recover the target word by another bi directional LSTM that is initialized by the current word state.

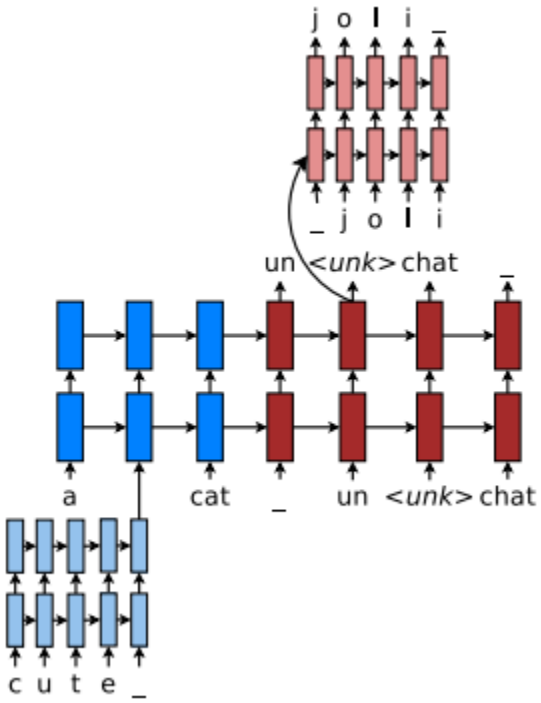

Figure 3.4 Hybrid Neural Machine Translation (M. Luong and C. D. Manning, 2016)

3.1.4 Human Evaluation

Human evaluation is the first method to evaluate fluency, grammar, coherence and fluency of the system. There are automatic ways of evaluation which are discussed in this chapter, section 3.1.5, however there are multiple alternative correct outputs for each sentence as input. The problem with human judges is their dependence on the human labour which is a time-consuming process and therefore, it is not efficient.

3.1.5 Automatic Question Generation Evaluation

Automatic evaluation metrics such as BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005) and ROUGE (Lin, 2004) are commonly used to measure the performance of generated questions by NQG systems.

(1) BLEU Measure

BLEU (Bilingual Evaluation Understudy) measure (Papineni et al., 2002) is widely used to evaluate machine translation systems. This metric is shown to be as one of the first measures to evaluate the correspondence between human and system's output (Bojar, Kos, and Mareček, 2010). BLEU utilizes modified form of the precision to compare the machine generated translation with multiple reference translations. If the generated text is close to any one of the references provided by humans, the BLEU score would be higher. BLEU is the averaged percentage of n-gram matches between the reference sentence and the hypothesis ($\hat{Y}$). BLEU n-gram score (p_n) on generated sentences (gen) is computed as:

$$p_n = \frac{\sum_{\hat{Y} \in gen} \sum_{n-gram \in \hat{Y}} Count_{clip}(n-gram)}{\sum_{\hat{Y}' \in gen} \sum_{n-gram \in \hat{Y}'} Count(n-gram')} \tag{3.2}$$

where $Count_{clip} = min(Count, Max\_Ref\_Count)$. $Count$ is the number of n-grams in $\hat{Y}$ and Max_Ref_Count demonstrates the maximum number of n-gram occurrences in the reference.

To compute the BLEU score, a **Brevity Penalty (BP)** term is used to penalize the score for short generated translations. If the length of the generated sentence is short, it is more likely for the generated n-garms to appear in the reference sentences. The BP factor is derived as follows:

$$BP = \begin{cases} 1 & if \ \ \hat{Y}_{length} < ref_{length} \\ exp(1 - \frac{\hat{Y}_{length}}{ref_{length}}) & \text{otherwise} \end{cases} \tag{3.3}$$

where $\hat{Y}_{length}$ and ref_{length} are the length of the hypothesis and the test corpus' effective reference respectively.

The **BLEU-N score** is defined as:

$$BLEU\_N = BP \cdot exp(\frac{1}{N} \sum_{n=1}^{N} logp_n) \tag{3.4}$$

(2) ROUGE Measure

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) measure (Lin, 2004) is commonly used to measure the performance of machine generated summaries and translation in contrast to human generated outputs. **ROUGE-N** computes the recall of n-gram overlap between the generated output and the reference in a set of summaries:

$$ROUGE\_N = \frac{\sum_{\hat{Y} \in gen} \sum_{n-gram \in \hat{Y}} Count_{match}(n-gram)}{\sum_{\hat{Y} \in gen} \sum_{n-gram \in \hat{Y}} Count(n-gram)} \tag{3.5}$$

where $Count_{match}(n-gram)$ is the maximum number of co-occurring n-grams between the generated ($\hat{Y}$) and the set of reference summaries (gen). **ROUGE-S** Allows any pair of word in the sentence order with arbitrary gaps to be considered to measure the overlap between the reference and the generated output. This is called as the *Skip-Bigram Co-Occurrence* statistics. **ROUGE-L** compares the longest matching sequence of words between the generated and the reference outputs. **ROUGE-W** also considers the longest matching sequence of words, however this measure favours the matches that are consecutive.

(3) METEOR Measure

METEOR (Metric for Evaluation of Translation with Explicit ORdering) measure (Banerjee and Lavie, 2005) is originally used for machine generated translation evaluation compared to human reference. This measure first takes the uni-gram matches m between the machine and reference translation and calculates the precision P and recall R:

$$P = \frac{m}{u_h} \tag{3.6a}$$

$$R = \frac{m}{u_r} \tag{3.6b}$$

where u_h and u_r are the number of uni-grams in the machine generated output and the reference respectively. This matching mechanism is not only based on the exact matches but it considers stemming and synonym relations as well. The METEOR

score is calculated as:

$$METEOR = F_{mean} * (1 - p)$$
$$F_{mean} = \frac{10PR}{R + 9P} \tag{3.7}$$

where F_{mean} is the weighted harmonic mean between precision and recall. p is the penalty term proposed by (Banerjee and Lavie, 2005).

3.2 Transfer Learning in Natural Language Processing

Supervised deep learning requires a large amount of labelled data to build a good model. Such a model is expected to perform well on the task and the domain of the data that the model is trained on. However, acquiring training data is often labor-intensive and expensive. Thus, ideas of using data in a different but related domain (referred to as the source domain) have proposed to help learn a model for the target task or domain of interest to achieve a better generalization. Since some of the information such as linguistic representations, semantics, etc., are shared across the source and the target settings, the knowledge learned and transferred from the source domain can be adapted to the target domain and tasks. In other words, knowledge such as model parameters can be transferred to a new task or domain, where there is a very limited amount of labeled data available. Transfer learning is such a technique.

3.2.1 Definition of Transfer Learning

Following S. J. Pan and Q. Yang (2010), we state the definition of transfer learning. We first define *domain* and *task* terms. Domain $\mathcal{D}$ consists of feature space $\mathcal{X}$ and a marginal probability *P(X)*, $\mathcal{D} = \{\mathcal{X}, P(X)\}$, where $X = \{x_1, x_2, ..., x_n\} \in \mathcal{X}$, and x_i is the vector representation of the ith input data (X is a learning sample). Based on the above definition, domains are different if they have different feature spaces or different marginal probability distributions. Task $\mathcal{T}$ consists of label space $\mathcal{Y}$ and conditional probability $P(Y|X)$ which is usually learnt from training data containing pairs of $\{x_i, y_i\}$, where $x_i \in X$, $y_i \in Y$ and Y is the corresponding labels for the particular learning sample X, $\mathcal{T} = \{\mathcal{Y}, P(Y|X)\}$. Given a source domain $\mathcal{D}_\mathcal{S}$, a source task $\mathcal{T}_\mathcal{S}$, a target domain D_T and a target task $\mathcal{T}_\mathcal{T}$, the objective of transfer learning is to improve the learning of the target conditional probability distribution $P(Y_T|X_T)$ in $\mathcal{D}_\mathcal{T}$ utilizing the knowledge obtained from $\mathcal{D}_\mathcal{S}$ and $\mathcal{T}_\mathcal{S}$, where $\mathcal{D}_\mathcal{S} \neq \mathcal{D}_\mathcal{T}$, or $\mathcal{T}_\mathcal{S} \neq \mathcal{T}_\mathcal{T}$. Based on the definition of the domain if $\mathcal{D}_\mathcal{S} \neq \mathcal{D}_\mathcal{T}$, then either $\mathcal{X}_\mathcal{S} \neq \mathcal{X}_\mathcal{T}$ or $P_S(X) \neq P_T(X)$. Similarly, if $\mathcal{T}_\mathcal{S} \neq \mathcal{T}_\mathcal{T}$ then either $\mathcal{Y}_\mathcal{S} \neq \mathcal{Y}_\mathcal{T}$ or $P(Y_S|X_S) \neq P(Y_T|X_T)$.

3.2.2 Transfer Learning Scenarios

There are different scenarios where transfer learning can be applied (Ruder, 2017):

- The feature spaces of source and target domain are different, $\mathcal{X}_\mathcal{S} \neq \mathcal{X}_\mathcal{T}$, e.g., the languages used in both source and target domains are different, known as cross-lingual adaptation.

- The marginal probability distributions of source and target input data are different, $P(X_S) \neq P(X_T)$, e.g., the topics that are discussed in both source and target documents are different, that is known as domain adaptation.

- Source and target label spaces are different, $\mathcal{Y}_\mathcal{S} \neq \mathcal{Y}_\mathcal{T}$, e.g., label spaces for both source and target tasks are different.

- The conditional probability of source and target tasks are different, $P(Y_S|X_S) \neq P(Y_T|X_T)$, e.g., source and target documents are not balanced with respect to their label classes.

In this work, our NQG problem is defined to generate questions on the car manual topic, where there is not sufficient amount of labeled data available. Therefore, we study the effect of parameter initialization of the neural-based model trained on a large QA dataset (parent model) to our system (child model). The topics discussed in both datasets are different, consequently based on the definition of transfer learning $P(X_S) \neq P(X_T)$. We first train our parent model on the large available dataset, SQuAD, and then use the learnt parameters from the parent model to initialize the child model with the knowledge obtained in the previous step. The parameters of the model are further fine-tuned on the target training data.

3.2.3 Transfer Learning for NLP

Transfer learning can benefit the performance of many NLP algorithms since attributes such as linguistic information are shared across different NLP tasks. Moreover, gathering labelled data can be rather expensive, depending on the task. There-

fore, transfer learning would assist such settings. Various transfer learning methods have improved the performance of NLP tasks (D. Wang and Zheng, 2015). Pre-training approaches in NLP are mostly based on the language modeling task. Pre-trained word embeddings such as Word2vec (Mikolov, Sutskever, et al., 2013) and GloVe (Pennington, Socher, and C. Manning, 2014), etc., are key factors for language understanding and training various NLP models. These are word-level approaches that have shown improvements in the performance of target models (Collobert et al., 2011; Xiong, Zhong, and Socher, 2016). Skip-Thought vectors introduced by (R. Kiros et al., 2015) are derived from an encoder-decoder framework that attempts to predict the surrounded sentences in a passage in an unsupervised matter. The intuition of this work is that consecutive sentences can be useful to construct sentence representation. R. Kiros et al. (2015) have shown good performance on 8 transfer tasks using these vectors. Universal sentence representations by (Conneau et al., 2017) outperformed the existing unsupervised approaches. Peters, Neumann, et al. (2018) proposed new word representations learnt by training a deep bi directional language model. The word vectors in this proposed method are a function of the entire sentence. They showed using these new representations would benefit the performance of NLP tasks such as question answering, textual entailment, semantic role labeling, coreference resolution, named entity extraction and sentiment analysis. Howard and Ruder (2018), Radford (2018), and Devlin et al. (2018) also presented effective transfer learning approaches where pre-trained embeddings could be fine-tuned on the available labelled training data in many NLP tasks. Recently, the pre-trained models are trained on deeper models (e.g., 24 layers (Devlin et al.,

2018)) in contrast to earlier approaches ((Yoshua Bengio et al., 2003) with 1 layer). In order to use the pre-trained models on downstream tasks, approaches such as feature extraction and fine-tuning can be applied. Feature extraction is referred to using the parameters of the pre-trained model on the target task (i.e., the parameters are frozen and transferred to another model). However, the fine-tuning approach optimizes the parameters of the pre-trained model on the target task. The model would converge faster in contrast to random initialization. Features from the pre-trained models can be transferred and fine-tuned using various training objectives (Felbo et al., 2017; Long et al., 2015). Transfer learning has also shown huge success in computer vision, where deep neural networks trained on ImageNet (Russakovsky et al., 2014) appear to be successful as feature extractors for multiple tasks such as image captioning (Karpathy and Li, 2014) and visual question answering (Hu et al., 2015). In Natural Language Processing, tasks such as sequence tagging and named entity recognition benefited from transfer learning as well (Z. Yang et al., 2015; Chiticariu et al., 2010). Chung, H. Lee, and Glass (2017) experiment the performance of two models on TOEFL listening comprehension test (Tseng et al., 2016) and MCTest (Richardson, Burges, and Renshaw, 2013) via a transfer learning method from MovieQA dataset (Tapaswi et al., 2015) for a multi-choice question answering task. By pre-training the model on MovieQA the performance of models rose considerably. Mou et al. (2016) demonstrate that transfer learning on NLP models relies on how source and target tasks are similar to each other semantically.

We observe two major issues with the exiting NQG models: (1) Unknown words that appear by shrinking the vocabulary size to reduce the computational cost during

test and training end-to-end deep models, are represented by one input vector. Hence, the semantic aspect of the rare/unknown words would not be captured by the model. (2) Training neural-based approaches for QG systems requires a large amount of training data. This amount might not be available for a specific domain of study. Therefore, it would lead deep neural networks to overfit. In our work, we address these issues in the context of NQG: (1) In order to solve the problem with the small available labeled QA dataset on a specific domain of study, we investigate the impact of parameter initialization from the model trained on an existing large dataset, SQuAD (Rajpurkar et al., 2016). The parameters are transferred to a model to be trained on a small set of labeled examples; (2) To address the problem with rare and unknown words, we integrate the relationships in the WordNet lexical database (Miller et al., 1990) into the NQG system. In particular, we make use of synonym connections defined in WordNet to help compute the input representations of unknown words.

Chapter Four

Improving NQG with Transfer Learning and Concept-Aware Word Embeddings

In this chapter, we first explain the architecture of our baseline model, Maxout Pointer and Gated Self-attention Networks (MP-GSN) (Zhao et al., 2018), that is originally proposed for the neural question generation task. Secondly, we investigate the effect of transfer learning on this task and improve our baseline model on a domain where available training data are not sufficient. Thirdly, we introduce the concept-aware model to incorporate the general knowledge of human beings into the input representation of our NQG system. Finally, we explain our experiments and give a brief summary of our proposed methods.

4.1 Baseline (Maxout Pointer and Gated Self-attention Networks)

Given an input answer $X = \{x_1 , x_2, ... , x_M\}$, the task of our NQG system is to generate the target question $Q = \{q_1 , q_2, ... , q_T \}$ where x_i and q_i are words, and M and T are the length of input and output sequences, respectively.

Our system is based on Paragraph-level Neural Question Generation with Maxout Pointer and Gated Self-attention Networks (MP-GSN) introduced by Zhao et al. (2018). The MP-GSN model was designed to tackle the problems brought by long text as inputs by using gated self-attention encoder and maxout pointer decoder. The model has shown great performance with both sentence-level and paragraph-level inputs. The architecture of the MP-GSN model is depicted in Figure 4.1. The details of the model are described as follows.

(1) Encoder with Self-attention

The encoder in MP-GSN is a two-layer bidirectional LSTM. As shown in Figure 4.1, in the paragraph-level of MP-GSN at each time step of the encoder, the previous hidden state and the concatenation of the word embedding e_t and meta-word representation m_t of whether the corresponding word is in or out of the answer are fed into the LSTM. However, in the sentence-level approach of MP-GSN we only feed the word embedding into the encoder.

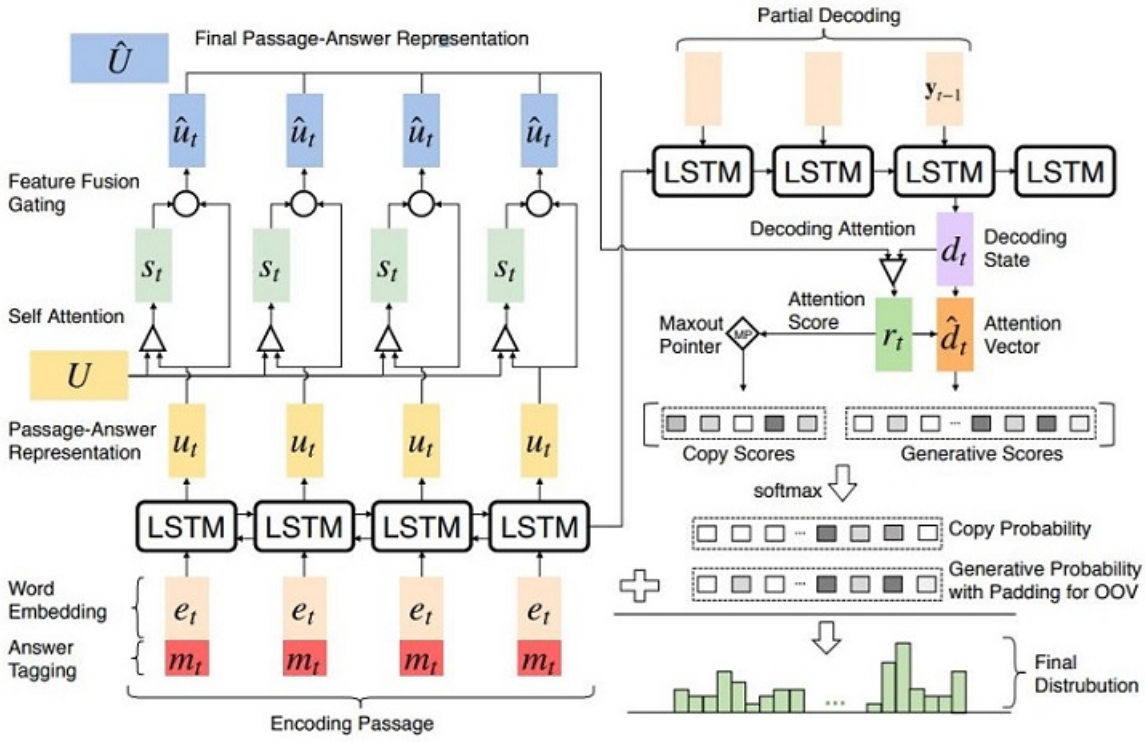

Figure 4.1 Maxout Pointer and Gated Self-attention Networks for NQG by (Zhao et al., 2018)

$$\overrightarrow{u_t} = \overrightarrow{LSTM}(w_t, \overrightarrow{u_{t-1}})$$
$$\overleftarrow{u_t} = \overleftarrow{LSTM}(w_t, \overleftarrow{u_{t-1}})$$
$$u_t = [\overrightarrow{u_t}, \overleftarrow{u_t}]$$
$$U = [u_t]_{t=1}^{M} \tag{4.1}$$

where u_t represents the RNN hidden state at step t and it is the concatenation of $\overrightarrow{u_t}$ and $\overleftarrow{u_t}$, and w_t is the embedding of the word at step t of the input text.

A gated self-attention mechanism (W. Wang et al., 2017) is applied to the encoder

to capture dependencies within the passage hidden states and therefore improve the encoded representation:

$$\alpha_t = softmax(UW_u u_t)$$
$$s_t = \sum_{t'=1}^{M} \alpha_t u_{t'} \tag{4.2}$$

where W_u is learnable parameter, α_t is the attention weight between u_t and elements in U, and s_t is the self-matching representation vector for u_t which is the weighted sum of the input hidden states. The final passage representation $\hat{U}$ is computed as follows:

$$f_t = tanh(W_f[s_t; u_t])$$
$$g_t = \sigma(W_g[s_t; u_t])$$
$$\hat{u_t} = g_t \cdot f_t + (1 - g_t) \cdot u_t$$
$$\hat{U} = [\hat{u_t}]_{t=1}^{M} \tag{4.3}$$

where W_f and W_g are learnable parameters, f_t is the new enhanced self-matching representation at step t which is derived by concatenating s_t and u_t, g_t decides the amount that u_t is updated, and $\hat{u_t}$ is the final encoder state at step t.

(2) Decoder with Maxout Pointer

The decoder is a two-layer unidirectional LSTM. To compute the hidden state s_t at step t, the decoder receives the word embedding w_t and the previous decoder state s_{t-1}. The decoder hidden state is used to calculate the attention distribution

α_t:

$$d_t = LSTM(w_t, d_{t-1})$$
$$\alpha_t = softmax(\hat{U} W_d d_t) \tag{4.4}$$

where W_d is the learnable parameter. The weighted sum of encoder hidden states (context vector c_t) is calculated by the attention distribution for time step t of the decoder:

$$c_t = \sum_{i=1}^{M} \alpha_{t,i} u_i \tag{4.5}$$

The context vector c_t is concatenated with the decoder state s_t and fed through the tanh layer to compute the new decoder state ($\hat{s}_t$):

$$\hat{d}_t = tanh(W_{\hat{d}}[d_t, c_t]) \tag{4.6}$$

where $W_{\hat{d}}$ is the learnable parameter. The generative score at step t is computed using equation (4.7). This score is utilized to produce the probability of sampling a word from the vocabulary, P_{vocab} in equation (4.8). P_{vocab} is used to predict words w in equation (4.9).

$$score_t^{gen} = W_v \hat{d}_t \tag{4.7}$$
$$P_{vocab} = softmax(score_t^{gen}) \tag{4.8}$$
$$P(w) = P_{vocab}(w) \tag{4.9}$$

where W_v is the learnable parameter. In MP-GSN, the copy mechanism is similar to the one in (Gu et al., 2016). In order to avoid repetitions in the output sequence, especially when the input sequence is long, Zhao et al. (2018) proposed a maxout pointer mechanism to calculate the score of copying word w from the input sequence at step t:

$$score_t^{copy}(w) = \begin{cases} \underset{i:w=x_i}{max}\ \alpha_{it} & x_i \in X \\ -inf & \text{otherwise} \end{cases} \tag{4.10}$$

Finally the concatenation of $score_t^{copy}$ and $score_t^{gen}$ is passed to a *softmax* function and probabilities pointing to the same words are summed up to generate a new word. Zhao et al. (2018) conducted experiments on SQuAD (Rajpurkar et al., 2016) and MS MARCO (Nguyen et al., 2016) datasets on both paragraph-level and sentence-level inputs. Their proposed method improves the generation of the questions in terms of BLEU_1, BLEU_2, BLEU_3, BLEU_4, METEOR and ROUGE-L.

4.2 Transfer Learning for Neural Question Generation

As part of our collaborative project with iNAGO Inc.[1], our NQG system is designed to generate questions from car manuals. The provided dataset in the specific domain of *Car Manual* consists of 4,672 question-answer pairs, which is not sufficient to train a deep neural question generation model. The performance of supervised

[1]http://www.inago.com

machine learning methods relies on the amount of available labelled training data. They are prone to overfit when there is not sufficient training examples available. To overcome this issue, we apply a transfer learning method. Fine-tuning is one of the transfer learning techniques where a model that has been trained on a task is transferred to a second similar task by initializing the second model with the parameters learned for the first task and fine-tuning the model with the data for the second task. This would allow the second model to benefit from the feature extraction from the previous model instead of training the parameters from the scratch.

We investigate three issues in the application of the transfer learning to neural question generation. First, we would like to see whether using a large amount of question-answer (QA) pairs from a general domain to train the first model is beneficial for initializing the model for the second task. Second, we would like to investigate whether transferring only parts of the parameters in the first model is beneficial to the second task. Finally, since the QAs in the general domain may not be relevant to car manuals, we would like to investigate whether selection of more relevant QAs from the first data set to train the first model would be beneficial to the second task. The diagram for the proposed approaches are represented in Figure 4.4.

4.2.1 Transfer Learning of the Whole Model with All Data

We investigate how fine-tuning of the whole first model generated with all the data in the first task would address the deficit of data in our NQG system. We train our model in two stages. In the first stage, we train an NQG system (that is, an MP-

GSN model[2]) on a large amount of question-answer (QA) pairs. The SQuAD dataset (Rajpurkar et al., 2016) is used for such a purpose, which contains approximately 80k QAs generated from Wikipedia pages.

In the second stage, we fine-tune the NQG model trained in the first stage on the car manual dataset. That is, instead of random initialization in training the second model, we utilize the parameters learnt from the SQuAD dataset to initialize the second model, and then fine-tune the initialized parameters on the dataset in the car manual domain.

Based on our experiments, shown in Chapter 5, transferring learnt parameters using the big QA dataset (SQuAD) proved to be advantageous and leads to a better model in cases where there is a small amount of labeled data available. As shown in Table 5.1, this technique could improve BLEU-1, BLEU-2, BLEU-3, BLEU-4, METEOR and ROUGE-L considerably.

4.2.2 Partial Model Transfer

We further investigate the impact of partially transferring parameters from different layers of the source model. We study which part of the knowledge can be transferred from the parent model to the child model to make improvements to the performance of the target and to obtain a black-box understanding of transfer learning. In all the experiments we use the pre-trained GloVe word embeddings. In this investigation, we divide the parameters of our model into two sets: (1) embedding

[2]In both phases of training an MP-GSN model, we use the GloVe embeddings for the input words, and freeze the word embeddings during the training.

parameters and (2) non-embedding parameters (which we refer to as inner-layer parameters). We would like to see whether transferring only the embedding parameters or the ones in the inner layers would increase or decrease the performance of the target model. When we transfer only the embedding parameters, we randomly initialize the inner-layer parameters of the child model, and the embedding layer's weights are initialized from the parent model,while they are freezed throughout the fine-tuning step. Then, inner layer's parameters are trained on the target domain.

When transferring only the inner-layer parameters, we use pre-trained GloVe embedding as the word representation for our target model vocabulary. Inner layer parameters are transferred from the source/parent model. The embedding weights are freezed, while other parameters are fine-tuned on the target model. We experiment how fine-tuning this pre-trained model on the child model effect the performance of the system. We observe minor improvement when the pre-trained word embeddings are updated on the target domain.

We will show in Table 5.2 that transferring either the embedding or inner-layer parameters benefits the second model, with the inner-layer parameter transfer producing a better target model. In addition, knowledge gained from both embedding and inner layers of the pre-trained model improves our target model the most significantly. The best performance is achieved when inner layers parameters are transferred and the pre-trained GloVe embeddings are fine-tuned along with the embedding and inner layer's parameters.

4.2.3 Data Selection for Transfer Learning

In this section we investigate the impact of the similarity between the data in the source domain and the data in the target domain on the transfer learning performance in the NQG task. As mentioned earlier, our source data set, SQuAD, is a general-purpose QA dataset containing questions generated from Wikipedia pages that cover various topics. The QAs in SQuAD are very different from the QAs in our target car manual data set. However, some QAs in the SQuAD data set may be more similar or relevant to the car manual domain than others.

In this investigation, we explore the effect of selecting from the SQuAD data set instances that are more relevant to the target domain of car manuals and removing less relevant examples from the source domain on the performance of transfer learning. For such a purpose, we explore two different ways to measure the similarity between an instance (i.e., a QA pair) in SQuAD and the car manual domain. Both similarity-measuring methods are based the cosine similarity measure, but they use different text representations.

(1) Similarity Measure Based on Sentence-level Representation

In this method, for each instance x_i^s in the source (e.g., SQuAD) data set (where s denotes *source* and i is the instance index in the source data set), we compute a similarity score between x_i^s and each instance x_j^t in the target (e.g., car manuals) training set (where t denotes *target* and j is the instance index in the target data set).

To calculate the similarity, each instance is represented by a vector. We use

the pre-trained BERT language representation model (Devlin et al., 2018) to compute the vector for each instance. As described in Section 2.1.6, the BERT model was trained on BooksCorpus (Zhu et al., 2015) and English Wikipedia passages by jointly conditioning on both left and right context in all layers on two tasks: masked language model and next sentence prediction. The trained model can provide a contextualized embedding for an input text. To obtain an embedding for an instance in either of our data sets, We feed the instance (i.e., a question and paragraph/sentence pair) to the pre-trained BERT model and retrieve a vector representation for the input.

The similarity between instance x_i^s in the source data set and instance x_j^t in the target training set is then computed by the cosine similarity measure:

$$CosineSim(E_i^s, E_j^t) = \frac{E_i^s \cdot E_j^t}{\|E_i^s\| \cdot \|E_j^t\|} \tag{4.11}$$

where E_i^s and E_j^t are the vector representations of x_i^s and x_j^t, respectively.

Then, the similarity between x_i^s and the target domain is measured by the average similarity between x_i^s and each instance in the target training data, as follows:

$$SimilarityScore(x_i^s) = \frac{\sum_{j=1}^{N} CosineSim(E_i^s, E_j^t)}{N} \tag{4.12}$$

where N is the number of instances in the target training data set.

We keep an instance if its similarity value is above a pre-defined threshold. In our experiments, we set different threshold values to investigate the effect of source data selection on the transfer learning performance.

(2) Similarity Measure Based on Word-level Representation

In this method, we use a word-level similarity to compute a a similarity score between an instance x_i^s in the source data set and the target domain. For this purpose, we use a set of keywords to represent the target domain. The set of keywords for the car manual domain is obtained using the LDA topic modeling technique (Blei, Ng, and Jordan, 2003) that identifies topics from input documents (which are car manuals in our case) [3]. In topic modeling, each topic is a distribution over words. We choose the top 10 words from each topic and merge them into a list of keywords. Furthermore, we use TopMine (El-Kishky et al., 2014) to extract top phrases. TopMine is an LDA-based method that involves first mining frequent phrases, and then using these phrases to represent documents. We select the resultant top phrases and break them down into unigrams and add them to the list of keywords. The list of the keywords are provided in the supplementary material.

To measure the similarity between an instance x_i^s in the source data set and the target domain, each word in x_i^s and the target domain keyword set is represented by its pre-trained GloVe embedding (Pennington, Socher, and C. Manning, 2014). For each word w in x_i^s, we compute the cosine similarity between w and each word in the target domain keyword set, and then average the similarity scores over all the words in the keyword set to obtain the similarity between w and the target domain. Then, the similarity between x_i^s and the target domain is the average of such similarity scores among all the words in x_i^s. Finally, an instance in the source data set is selected if its similarity score to the target domain is above a user-defined threshold.

[3]The number of topics is set to be 20, since this number results in the lowest perplexity score

4.3 Concept-aware Model for Neural Question Generation

The problem of rare and unknown words can affect the performance of NLP systems and cause negative impacts on understanding the natural language. Recent NLP-based approaches use softmax as the output layer of their system. The output dimension of this layer corresponds to the vocabulary size. This vocabulary is usually shrunk to contain the top-k most frequent words from the training data. This would make training deep neural networks computationally less expensive. The words that do not appear in this vocabulary are unknown words and they are represented by $< unk >$ token. This representation would bring issues such that the system is not able to learn good representation of the infrequent words occurring during the training step. Furthermore, the key information in the text representation can be lost due to the replacement of these words with only a single token during training and test time (T. Luong, Sutskever, et al., 2014).

There are approaches to tackle this matter as discussed in section (3.1.3), such as pointer softmax (Gülçehre et al., 2016) in machine translation and text summarization, character-level embeddings (Ling et al., 2015) in language modeling, part-of-speech tagging, etc. To address the rare and unknown words problem for neural question generation task, we propose a concept-aware model. Inspired by (F. Liu et al., 2019), a model that is proposed to handle unknown words in machine translation systems, we present concept-aware model for the neural question generation problem. We encode enhanced vector representations for words that are

not seen throughout the training and test steps. WordNet (Miller et al., 1990), a lexical database containing semantic relations between words in English, assists our NQG system in generating these representations of rare and unknown words. In other words, we incorporate our NQG system with the general knowledge of human beings. However, unlike (F. Liu et al., 2019) by which sense embeddings are derived through LSTM-RNN language model, we obtain word embeddings from GloVe. We look for the words in the embedding table of the system. Also, the sense vector representation from synonyms are derived by computing the average over the pre-trained word embeddings. Instead of using the last hidden state as the sentence embedding, we sum over the word vectors in a sentence and divide by the number of the words in each sequence.

4.3.1 WordNet Lexical Database

Wordnet (Miller et al., 1990) is an on-line lexical database in English developed in Princeton University by Cognitive Science Laboratory [4]. It is used in many natural language processing systems such as information retrieval, word sense disambiguation (this task determines the sense of the word in a specific context), etc. Words are grouped into *synsets* in this database. The term synset or synonym set is referred to elements that are semantically similar to each other. In other words, words having the same sense (a *word sense* is one aspect of the meaning of a word) are grouped into a synset. If a word has multiple senses, then it belongs to different synsets. Figure 4.2 shows senses of the word *bass* with their meanings in this lexical database

[4]http:// www.cogsci.princeton.edu/

[5].

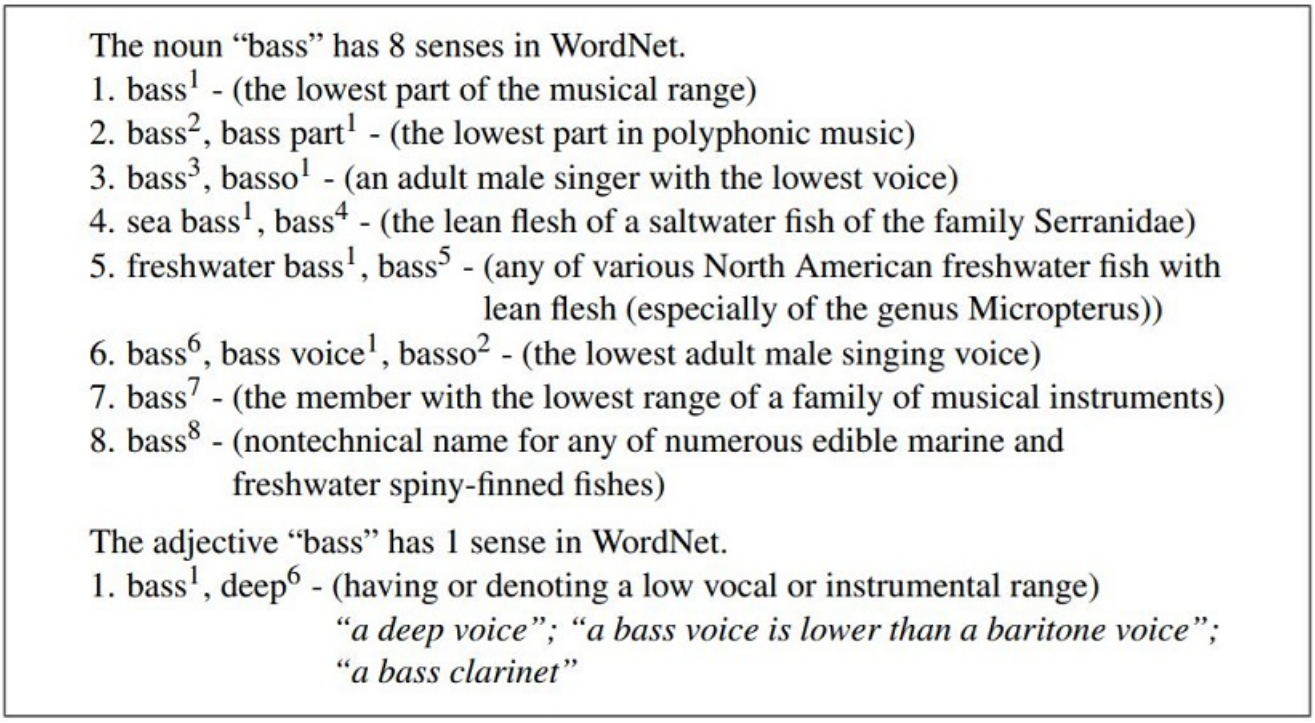

Figure 4.2 A portion of the WordNet 3.0 entry for the noun *bass*

WordNet groups words by lexical and semantic relations. These relations are synonymy, hyponymy, hypernymy, meronymy and etc. Hyponym refers to a word whose meaning is included in the meaning of another word, while hypernym is a word that its meaning include a group of other words. Meronymy is the relation which indicates a member of something. Figure 4.3 is an example of the representation of semantic relations between words. NLTK (Bird and Loper, 2004) is a toolkit for natural language processing systems in English. Definition, examples and relations for each synset can be obtained from this toolkit. Table 4.1 shows an example from

[5]https://web.stanford.edu/ jurafsky/slp3/C.pdf

WordNet using NLTK for the noun "*internet*" which has 1 sense in this database.

Word: Internet

Synsets: Synset('internet.n.01')

Definition: a computer network consisting of a worldwide network of computer networks that use the TCP/IP network protocols to facilitate data transmission and exchange.

Synonyms: 'net', 'cyberspace'

Table 4.1 Definition and synonyms for the word *"internet"* in the WordNet lexical database.

4.3.2 Integrating WordNet relation into neural question generation system

To deal with the unknown word problem for neural question generation, we exploit synonym relation defined in WordNet into our system. We capture a better understanding of the natural language when encountering unknown and rare words. Unknown words are usually converted to $< unk >$ token in NLP systems during both training and testing. Instead of simply converting these words to a single token and having one vector representation throughout training and inference phase, we embed the conceptual aspect of the word using WordNet. We incorporate the neural networks of question generation with the knowledge of human beings. Having constructed a vector for each word that is not in the vocabulary list and would be

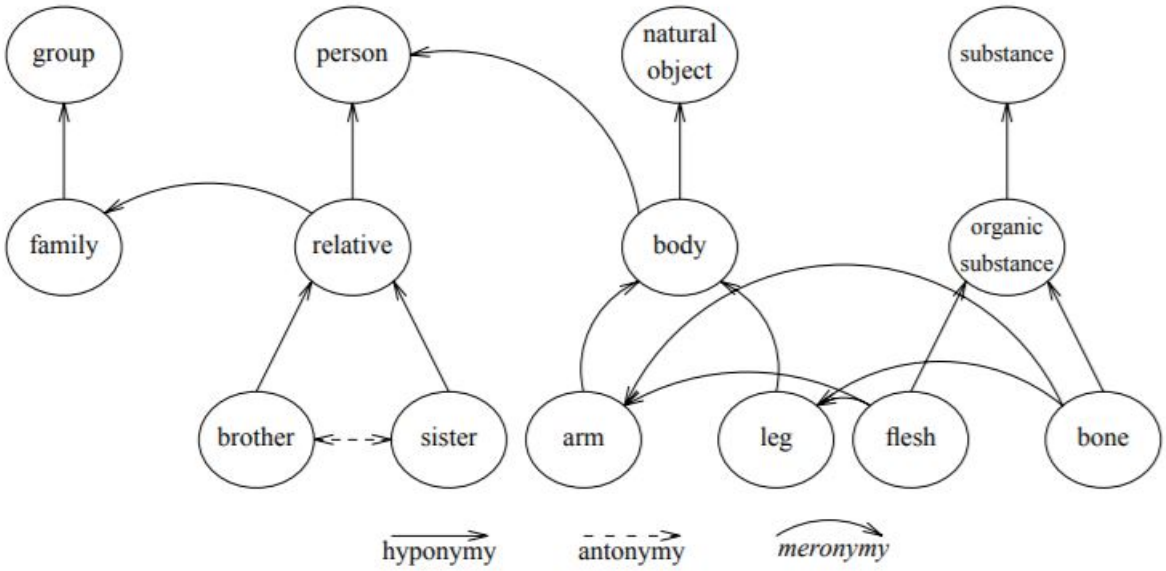

Figure 4.3 Network representation of semantic relations (Miller et al., 1990)

mapped to the $< unk >$ token in the embedding layer, the system have a better understanding of the input text in terms of semantics. We further demonstrate the effectiveness of this method for our NQG system when the vocabulary size is small due to short amount of labeled training data being available in a specific domain of study. The details of our proposed concept-aware model is as follows.

For each unknown word x_m, we obtain a set of synonyms $S_m^{'}$ from WordNet, where $S'_m = \{s'_1 , s'_2, ... , s'_k\}$ and s'_i is the ith synonym for x_m and k is the number of obtained synonyms. The synonyms in set S'_m are represented by deriving their vector representations from the embedding table of the system. The embedding table contains the word indices mapped to their corresponding distributed representations. The vector representations in the table can be randomly initialized and learnt during

training. We choose pre-trained GloVe (Pennington, Socher, and C. Manning, 2014) word embeddings to initialize the word vector values. Instead of having only one representation for the $< unk >$ token, we compute a conceptual representation based on elements in S'_m. We search for the word embedding of each synonym s'_i in the embedding table and call this $emb(s'_i)$. Assuming p as the number of existing synonyms in the embedding table, where $0 \leq p \leq k$. Then, we compute the conceptual vector for the unknown word x_m as follows:

$$emb^*(x_m) = \frac{emb(s'_1) + emb(s'_2) + ... + emb(s'_p)}{p} \tag{4.13}$$

where $emb^*(x_m)$ is the *concept-aware* vector for x_m. The fixed representation of $< unk >$ token for the unknown word x_m is replaced by the new enhanced vector $emb^*(x_m)$ in the embedding layer and is fed into the encoder during both training and test time. For example, as shown in Table 4.2 our model found the embedding vectors of *"documentary", "docudrama" and "documentary-film"* (synonyms for the unknown word *"infotainment"*) from the embedding table. Then, the concept-aware vector is computed based on the synonym representations as in equation (4.13) and fed into the encoder of the system. Therefore, we incorporate the knowledge of human beings into the neural based approach for QG and capture a better understanding of the input sentence. If $p = 0$, we use the $< unk >$ representation as the word embedding.

In cases where for a specific unknown word, multiple synsets are captured, we apply the following steps:

- We first compute an enhanced conceptual vector for each synset based on its synonyms as in equation (4.13) .

Unknown/Rare word	Synset	Synonyms
"slick"	*"slickness.n.03"*	*"slip"*
"consultant"	*"adviser.n.01"*	*"advisor"*

Table 4.2 Examples of unknown word synonym relations derived from the WordNet lexical database and the embedding table of our system.

- For each synset, we replace the $< unk >$ token representation with the concept-aware vector. Therefore, we construct a set of sentences $s = \{s_1, s_2, ..., s_m\}$ where m is the number of the synsets and each sentence contains a different conceptual embedding vector of the unknown word.

- For each sentence s_i in s, we compute a sentence embedding $emb(s_i)$ by averaging over its word vectors. Similarly, we calculate the original sentence embedding $emb(s^*)$ containing the $< unk >$ token.

- To choose one of the synsets we calculate the cosine similarity between each sentence embedding $emb(s_i)$ and $emb(s^*)$ as follows:

$$cos(\vec{emb}(s_i), \vec{emb}(s^*)) = \frac{\vec{emb}(s_i) \cdot \vec{emb}(s^*)}{\|emb(s_i)\| \|emb(s^*)\|} \tag{4.14}$$

- Among all subsets in s, we pick a sentence whose embedding makes the cosine similarity value higher. The word representation of the unknown word is derived from the selected sentence.

Finally, the enhanced word vector representations are used in the embedding layer and the model is trained using maxout pointer and gated self-attention networks, section (4.1).

4.4 Summary

In this chapter, we have presented a concept-aware model to address the problem with rare and unknown words and investigated the effect of transfer learning on the neural question generation problem. Figure 4.4 illustrates the structure of the presented approaches and contributions of the thesis.

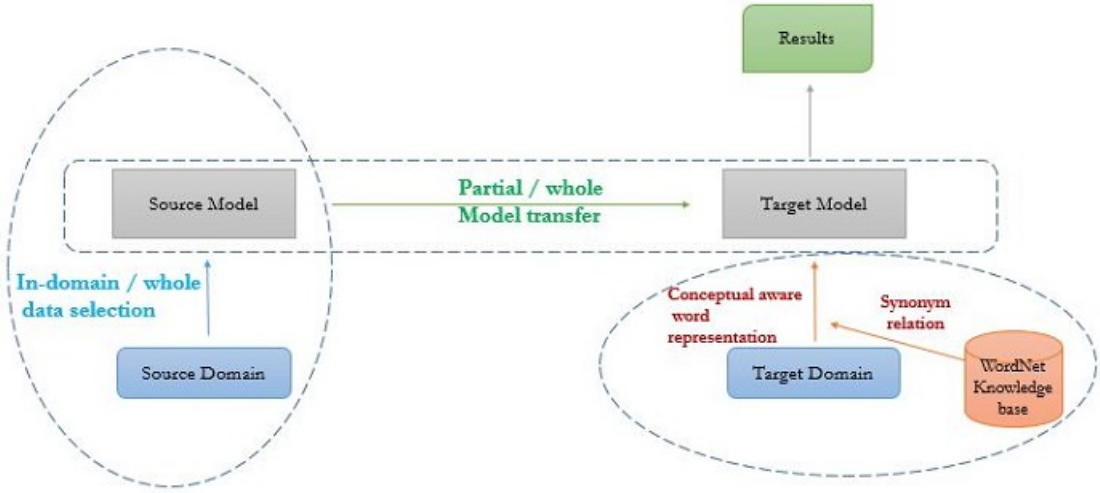

Figure 4.4 Transfer Learning and Concept-aware approaches for Neural Question Generation.

Chapter Five

Empirical Evaluation

This chapter provides information on the settings of our experiments, description of the datasets and evaluation metrics that are used. We further report and discuss the outcome of each experiment. In the end, we provide examples of the outputs generated by our models.

5.1 Datasets

- **Car Manual dataset:** This dataset (provided by iNAGO Inc. [1]) contains 4672 questions and answers (QAs) created by human annotators from two car manuals of Ford and GM, denoted as Car Manuals. We divide the dataset into training (4360 QAs) and testing sets (312 QAs).

- **SQuAD dataset:** We use the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016). This is a reading comprehension dataset that contains questions posed by crowdworkers about the Wikipedia articles. The answers to

[1] http://www.inago.com

questions are obtained from a segment of text from the articles. We adopt types of SQuAD: 1) Paragraph-level: The dataset is split into training set (87,599 QA pairs), a validation set (5,285 QA pairs) and test set (5,285 QA pairs). Answers to the questions are specified by BIO (Inside-outside-beginning) tagging. 2) Sentence-level: We use the processed SQuAD (Du, Shao, and Cardie, 2017). The data has been divided into a training set (70,484 QA pairs), a validation set (10,570 QA pairs) and test set (11,877 QA pairs).

5.2 Evaluation Metrics

We report BLEU-1, BLEU-2, BLEU-3, BLEU-4 (Papineni et al., 2002) , METEOR (Banerjee and Lavie, 2005) and ROUGE (Lin, 2004) scores as the evaluation metrics of our NQG system using the nlgeval (Sharma et al., 2017) package [2]. BLEU-N is a modified precision of n-grams between the generated questions by our NQG system and the reference questions. ROUGE-L compares the longest matching sequence of words between question generated by the system and reference questions. METEOR calculates the similarity between the generated and reference questions by considering paraphrases, synonyms and stemming. The details regarding the automatic evaluation metrics used in our experiments are discussed in chapter 2 (section 3.1.5).

[2]https://github.com/Maluuba/nlg-eval

5.3 Baseline Systems

- **Seq2seq + attn**: We implement a sequence-to-sequence model with attention mechanism as one of our baseline methods. We use the default setting as mentioned in (Huang, 2017a). We use gated recurrent unit (GRU) (Kyunghyun Cho et al., 2014) and the hidden size is set to 256. The model is trained using stochastic gradient descent optimizer with learning rate initialized to 0.01.

- **Transformer** (Vaswani et al., 2017): This model is originally presented for the sequence-to-sequence problem and it replaces the RNN-based models with the attention mechanism. The model is composed of two parts, Encoder and Decoder. Both Encoder and Decoder contain a stack of identical layers. The sequence can be processed in parallel during training. In the experiment, we use the default setting as in (Huang, 2017b).

- **MP-GSN** : This method is originally proposed for question generation which utilizes paragraph-level information with gated self-attention encoder and max-out pointer. A self-matching representation is first derived from the input as in (W. Wang et al., 2017). The maxout pointer mechanism avoids repetition during decoding. The method can do both sentence-level and paragraph-level question generation. Details of this model is discussed in section (4.1). In our experiment, we use pre-trained GloVe word vectors of 300 dimensions. We utilize Bidirectional LSTM encoder and unidirectional LSTM decoder both with hidden dimensions of 300. We set the number of layers to 2 in both the encoder and the decoder. We set the batch size to 16 due to the better performance

of the baseline model. We use the stochastic gradient descent (SGD) optimizer with learning rate 0.1 for 20 epochs. These are the default setting in the implementation of the algorithm in (S. Lee, 2019).

5.4 Experiments and Results on Whole Model Transfer and Concept-aware Generation

We first conduct the following set of experiments:

- seq2seq + attn on car manual: As our first experiment, we train and test this baseline model on the car manual training and test data.
- Transformers (Vaswani et al., 2017) on car manual: As the second experiment, we train and test this model on car manual dataset,
- MP-GSN (Zhao et al., 2018) on car manual: The MP-GSN is both trained and tested on the car manual dataset for this experiment.
- MP-GSN on SQuAD: As the forth experiment, we train the MP-GSN on SQuAD and test the model on the car manual dataset.
- Concept-aware model (section 4.3): The concept-aware MP-GSN is firstly trained on the car manual training data. Secondly, we use the car manual test data during prediction.
- Fine-tuning sentence-level SQuAD: In this experiment, we train sentence-level MP-GSN on SQuAD as the parent model, and the model is fine-tuned on car

manual training data and evaluated on car manual test data.

- Fine-tuning paragraph-level SQuAD: First, the paragraph-level MP-GSN is trained on SQuAD. Then, the parameters of the model is fine-tuned on the car manual training data. The performance of the model is evaluated on the car manual test data.

- Hybrid model: The last experiment is applied on our hybrid model by which both transfer learning technique and concept-aware model are combined. We train MP-GSN on the SQuAD as the *base* model. We train a concept-aware MP-GSN by initializing parameters from the *base* model. We test this model on the car manual dataset.

The results of our experiments are given in Table 5.1. Since during inference we use the car manual test data, we measure the evaluation metrics on sentence-level approach of MP-GSN. The MP-GSN which incorporates the gated self-attention encoder and maxout pointer decoder, outperforms the transformers and seq2seq + attn baseline models in terms of BLEU-1, BLEU-2, BLEU-3, METEOR and ROUGE-L on the car manual test dataset. The MP-GSN trained on SQuAD performs poorly during inference on the car manual domain test dataset. One possible explanation is the domain difference of the data during training and test time. Fine-tuning the sentence-level approach of MP-GSN trained using SQuAD on car manual training data (fine-tuning sentence-level SQuAD) would lead to performance improvements in terms of BLEU-2, BLEU-3, BLEU-4 and METEOR scores. However, by training the paragraph-level MP-GSN on SQuAD and further fine-tuning the model on the car

	Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L
Baselines	seq2seq+attn	32.63	17.51	10.49	6.64	12.32	32.29
	transformer	38.59	23.47	15.44	9.69	13.46	39.31
	MP-GSN (trained on carmanual)	58.16	46.74	39.34	33.65	29.49	59.15
Ours	MP-GSN (trained on SQuAD)	36.41	20.41	13.22	8.82	16.22	35.32
	concept-aware model	59.29	47.89	40.36	34.61	30.03	60.26
	fine-tuning sentence-level	57.7	47.09	40.74	35.66	30.04	59.06
	fine-tuning paragraph-level	**62.96**	52.24	45.34	39.68	32.65	**62.15**
	fine-tuning paragraph-level (+ GloVe)	**62.96**	**52.72**	**46.22**	**40.96**	**32.67**	62.03
	hybrid model	61.29	50.59	43.70	38.29	32.1	61.36

Table 5.1 Comparing the results of the baselines with the proposed approaches in terms of BLEU-1, BLEU-2, BLEU-3, BLEU-4, METEOR and ROUGE-L on the car manual test data.

manual training dataset (fine-tuning paragraph-level SQuAD), we achieve significant improvements in terms of all the evaluation metrics on the test dataset in compare to the baselines. This approach exceeds (Zhao et al., 2018) by (+4.8 BLEU-1, +5.5 BLEU-2, +6 BLEU-3, +6.03 BLEU-4, +3.16 METEOR, 3 ROUGE-L) points. We also explore the effect of fine-tuning the pre-trained GloVe word embeddings and observe improvements in terms of BLEU-1, BLEU-2, BLEU-3, BLEU-4, METEOR scores.

5.5 Comparison of Results between Partial and Full Model Transfer

We also investigate the effect of transferring different layers of our deep neural network model trained on large QA dataset (parent model) to the same architecture using car manual training dataset (child model). We divide the transfer learning technique into multiple steps: (1) transferring the embedding layer parameters, (2) transferring the non-embedding layers (inner layers) parameters, (3) transferring all the layers parameters. In the first experiment, embedding layer parameters in the source model are initialized by pre-trained GloVe word embeddings based on the SQuAD vocabulary set and they are transferred to the target model. As shown in Table 5.2, the embedding layer contains transferable information and it improves our baseline model by about 1 score in terms of all the evaluation metrics used in our work.

Moreover, we demonstrate the effect of transferring non-embedding layers (inner layers) using MP-GSN. During this experiment, we initialize the embedding parameters based on our target vocabulary set, and the inner layer parameters are transferred from the source model and further fine-tuned on the target domain. This approach increases the evaluation scores (+6.09 BLEU-1, +7.22 BLEU-2, +7.87 BLEU-3, +8.02 BLEU-4, +4.09 METEOR, +4.42 ROUGE-L) in comparison to our MP-GSN baseline model. This confirms that transferring the inner layer parameters plays the most crucial role of transfer learning on our NQG system.

In addition, transferring parameters from all the layers of the source model would

	Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L
	baseline	58.16	46.74	39.34	33.65	29.49	59.15
Ours	transfer embedding	59.67	47.60	40.19	34.54	30.22	60.58
Ours	transfer inner layers	64.25	53.96	47.21	41.67	33.58	63.57
Ours	transfer all layers	62.96	52.24	45.34	39.68	32.65	62.15
Ours	transfer all layers + fine-tuning GloVe	62.96	52.72	46.22	40.96	32.67	62.03
Ours	transfer inner layers + fine-tuning GloVe	**64.83**	**54.72**	**48.08**	**42.73**	**33.87**	**63.88**

Table 5.2 Exploring the model's performance by transferring different layers of the maxout pointer and gated self-attention network.

achieve better results in comparison to transferring only the embedding layer. We further demonstrate the effect of fine-tuning GloVe embeddings on the target domain in Table 5.2. Fine-tuning GloVe embeddings during the experiments slightly increases the BLEU, METEOR and ROUGE-L scores. The best performance during the analysis of the model transfer is achieved when the embedding table is initialized with the target vocabulary and inner layer parameters from the source model is fine-tuned on the target model without freezing the GloVe embeddings. We deduce that both embedding and non-embedding layers contain transferable information, but nonetheless they are not as optimal as transferring the inner layers parameters.

5.6 Results on In-domain Instance Selection

We further conduct experiments to see the impact of initializing parameters using instances that are more similar to the car manual domain. As described in Section

4.2.3, we use two types of similarity measures to measure the similarity of an instance in the source data set to the target domain to select instances from the source (i.e., SQuAD) data set: *sentence-level* and *word-level* similarity measures.

With the sentence-level based similarity measure, we extract subsets of SQuAD with the similarity value higher than pre-defined thresholds (i.e., 0.1, 0.2, 0.3 and 0.4), which result in $87,591$, $86,828$, $67,401$ and $14,518$ instances respectively. Consequently, the examples from SQuAD that are more similar to the car manual instances in terms of sentence vector representations are extracted. Note that the higher the threshold value is, the more similar the subset of selected instance is to the target domain, but the smaller the selected subset is. Since $87,591$ number of instances have similarity threshold more than 0.1 to our target domain, we conjecture that BERT embedding do not capture the good representation of the paragraph-level input.

With the word-level similarity measure, we select instances in the source domain data set whose similarity score is positive, which results in $14,090$ paragraph-question pairs to train the source model. The reason that we only use zero as the similarity threshold is that a higher threshold value (e.g., 0.1) leads to a subset of the source domain data that is smaller than our target domain data set.

The results of this experiment are displayed in Table 5.3. As shown in the table, by setting the threshold value to 0.4 for the sentence-level comparison or using the word-level similarity measure (with a threshold of zero), only 16% of the original source data are selected and used to train the source model, but the performance still rise considerably, compared with the baseline that does not use transfer learning.

However, neither of the methods outperforms the model trained by all the instances from SQuAD. The reason is that the size of the training data for both method is decreased significantly due to the selection. Thus, we conclude that the impact of having a large data set in the source domain outweighs using a smaller subset (about 16%) of the source data that are semantically more similar to the target domain. For the models with similarity thresholds of 0.1, 0.2 and 0,3, their performance is better than the baseline but again not as good as the model trained with all the source data. Although their data set sizes are not as small as the one when the threshold is set to 0.1, the sizes are smaller than the original. Since their similarity thresholds are low, we conjecture that the overall similarity of the selected data by these three thresholds to the target domain may not be much different from the one for the original data. and thus the size of the source training data is more important in this situation.

To see whether instance semantic similarity matters when the data sizes are the same or similar, we randomly select $14,575$ instances from the SQuAD data set to train the parent model, in order to compare with the model with the word-level similarity and the model with the 0.1 threshold for the sentence-level similarity. As shown in Table 5.3, we observe that using similar instances obtained from the word-level comparison results in a better performance in contrast to employing a randomly chosen subset of training data with the same amount of QA pairs. However, the performance of the model with the 0.4 threshold for the sentence-level similarity does not outperform the model with randomly-selected data set of similar size. We conjecture that the our sentence-level similarity measure might not well capture the

similarity between a training instance and the target domain. Based on our results, the word-level similarity measure (which is based on a set of keywords in the target domain generated by the LDA model) is better.

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L	# of examples
Baseline	58.16	46.74	39.34	33.65	29.49	59.15	0
(0.1 similarity threshold)	61.53	51	44.47	39.31	32.03	61.25	87,591
(0.2 similarity threshold)	61.37	50.79	44.27	39.13	31.6	60.71	86,828
(0.3 similarity threshold)	62.24	51.16	44.12	38.57	32.04	61.21	67,401
(0.4 similarity threshold)	61.06	49.73	42.60	36.98	31.42	60.94	14,518
(word-level similarity)	62.38	51.25	44.15	38.47	32.06	61.66	14,090
(samples randomly chosen)	61.19	49.85	42.74	37.12	31.31	61.08	14,575
(all the source data)	**62.96**	**52.24**	**45.34**	**39.68**	**32.65**	**62.15**	87,599

Table 5.3 Comparing effects of transfer learning on MP-GSN using different subsets of the source domain examples that are more similar to the target domain.

5.7 Discussion on Results for Concept-aware Model

As shown in Table 5.1, our concept-aware model could improve the MP-GSN model by at least about 1 point in terms of all the evaluation metrics. Our hybrid model which combines both fine-tuning and concept-aware word representation approaches does not demonstrate the most optimal performance. We believe this is due to deriving the concept-aware vector of each unknown word from a larger vocabulary set of $\approx$ 45K words during training and fine-tuning steps as opposed to the smaller vocabulary size of $\approx$ 5K using car manual QA pairs. Consequently, higher number

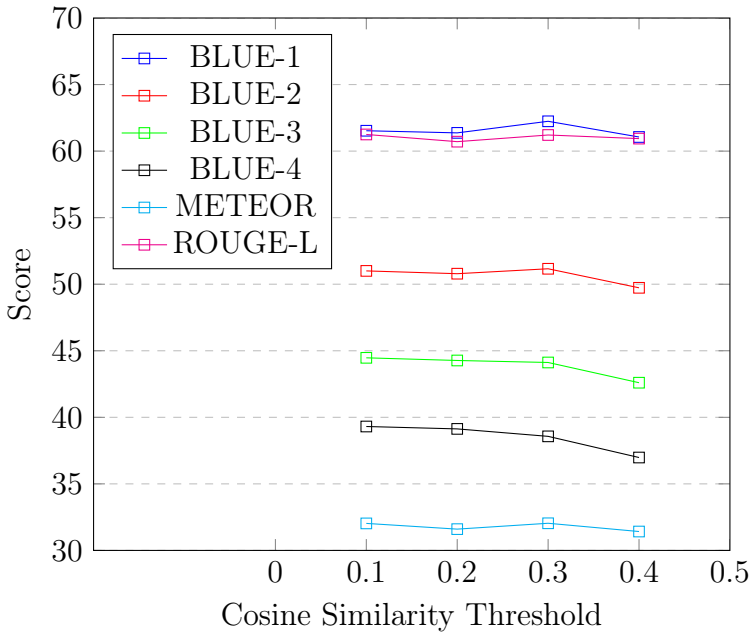

Figure 5.1 Comparing the performance of MP-GSN on the target domain by choosing in-domain instances from the source domain

of synonyms would match the existing words in the embedding table. The derived synonyms from WordNet might contain different senses, and therefore computing an average vector from their word vectors would not capture a good representation of the unknown word. On the other hand, by implementing the concept-aware model on the smaller QA dataset (consists of $\approx$ 5K words in the vocabulary set) in the domain of car manuals, synonyms related to this specific domain would be selected. Hence, a better representation for each unknown word is constructed, and this would result in a better performance in compare to our baseline model.

5.8 Examples

This section is devoted to examples from the test set, generated questions by our models along with the reference questions. In each example:

- **Sentence** denotes the source sentence from which the question would be generated.
- **MP-GSN** denotes the output question by our baseline model (4.1).
- **concept-aware model** represents the generated question by the concept-aware model (4.3).
- **Fine-tuned SQuAD** indicates the question generated after fine-tuning MP-GSN on car manual training data (4.2).
- **Reference** denotes the target question.

We observe that in in most examples MP-GSN struggles to generate grammatically correct questions or the question is not related to the answer provided. Concept-aware model and the transfer learning learning could improve the performance of the baseline model in terms of the aforementioned problems and in some cases these models are able to generate questions as well as human-written questions.

Example (1):

Sentence: if the vehicle is equipped , the position of the throttle and brake pedals can be changed.

MP-GSN: can i use the brake pedals be changed?

concept-aware model: can the position of the brake pedals be changed?

Fine-tuned SQuAD: can the position of the throttle and brake pedals be changed?

Reference: can the positions of the throttle be changed?

Example (2):

Sentence: the head-up display rotation option feature allows you to adjust the angle of the head-up display image.

MP-GSN: how does the head-up display rotation feature feature?

concept-aware model: how do i adjust the angle of the head-up display?

Fine-tuned SQuAD: what does the head-up display rotation option do?

Reference: what is the head-up display rotation option?

Continued on next page

Example (3):

Sentence: in canada , the law requires that forward-facing child restraints have a top tether , and that the tether be attached .

MP-GSN: what is the personalization options for forward-facing child restraints ?

concept-aware model: does the law requires that forward-facing child restraints have a top tether ?

Fine-tuned SQuAD: does the law need that child restraints have a top tether ?

Reference: what does canadian law say about forward-facing child restraints ?

Example (4):

Sentence: see my.cadillac.com to register your vehicle for the owner experience program.

MP-GSN: what should i do if the vehicle is program in the owner experience program?

concept-aware model: how do i use my vehicle for the owner experience program?

Continued on next page

Fine-tuned SQuAD: how can i register my vehicle for the owner experience program? **Reference:** how do i register my vehicle for the online owner experience program?
Continued on next page

Example (5):

Sentence: to enhance your ownership experience , we and our participating dealers are proud to offer courtesy transportation , a customer support program for vehicles with the bumper-to-bumper (base warranty coverage period in canada) , extended powertrain , and/or hybrid-specific warranties in both the u.s. and canada.

MP-GSN: what should i do if i am to enhance my ownership experience?

concept-aware model: how do i enhance my ownership experience?

Fine-tuned SQuAD: how do i enhance my ownership experience?

Reference: what is the courtesy transportation program?

Example (6):

Sentence: the proximity sensing option in the display feature allows the feature to be turned on or off.

MP-GSN: what is the proximity sensing option in the display?

concept-aware model: what is the proximity sensing option in the personalization menu for?

Fine-tuned SQuAD: what is the proximity sensing option in the display for?

Reference: what is the proximity sensing feature in the personalization menu for?

Continued on next page

Example (7):

Sentence: the roadside assistance program is not available for any of the countries in the central american and caribbean region.

MP-GSN: is the roadside assistance program available for?

concept-aware model: what is the roadside assistance program for?

Fine-tuned SQuAD: where is the roadside assistance program not available?

Reference: where is the roadside assistance program not available?

Example (8):

Sentence: replace wiper blades at least once per year for optimum performance.

MP-GSN: how can i replace wiper blades blades?

concept-aware model: how do i replace the wiper blades?

Fine-tuned SQuAD: how often should i replace wiper blades?

Reference: when should i replace wiper blades?

Continued on next page

Example (9):

Sentence: to access the driver door key lock cylinder , pull the door handle to the open position and hold it open until cap removal is complete. then , insert the key into the slot on the bottom of the cap , lift the key upward and remove the cap. finally , use the key in the cylinder.

MP-GSN: how do i access the driver door key?

concept-aware model: how do i access the driver door key lock?

Fine-tuned SQuAD: how do i access the driver door key lock cylinder?

Reference: how do i access the driver door key lock cylinder?

Example (10):

Sentence: operating the engine with the air cleaner/filter off can cause you or others to be burned.

MP-GSN: can i use the engine with the air with the air?

concept-aware model: can i use the engine with the air with the air?

Fine-tuned SQuAD: can i use the engine with the air cleaner off?

Reference: what can happen if i operate the engine with the air filter off?

Table 5.4 Examples of questions generated by MP-GSN, concept-aware model and fine-tuned MP-GSN on car manual dataset.

5.9 Summary

In summary, we tackle the issues brought by lack of labeled data and out-of-vocabulary words for training deep neural question generation model. We presented a concept-aware model to address the problem with out-of-vocabulary words and a transfer learning technique to solve the issue brought by having small labelled training examples. Our proposed models generate better quality questions in terms of grammar, coherence and their relatedness to the input answer by incorporating the existing large QA dataset i.e., SQuAD and integrating the word relations defined in WordNet. The experiments show that our proposed methods outperformed the state-of-the-art approaches for NQG in the specific domain of car manuals.

Chapter Six

Conclusion & Future Work

6.1 Conclusion

We present a concept-aware model and incorporate a transfer learning approach for neural question generation problem in this thesis. This task aims to automatically generate questions from text documents. The proposed methods address issues brought by having insufficient labelled training data, and encountering rare/unknown words problem during training and test steps for the NQG problem. We investigate the impact of transfer learning where the current NQG approaches suffer from the lack of a large amount of labelled training data in a specific domain. We also explore the effects of transferring different layers of the network for low-resource question generation. The results further highlights the use of model initialization for the NQG problem. As an another investigation, we evaluate our models trained on different subsets of the source training data using similarity measures. Sentence-level and word-level representations of the text are

The concept-aware model is introduced to solve the problem with rare/unseen

words. It integrates semantic relationships defined in the WordNet lexical database, which is a type of general knowledge external to the training data, into the input representation of our NQG system. Our results indicate the effects of using the parameter initialization method as well as using external knowledge into our QG model, and achieve a better performance across various automatic evaluation metrics, such as BLEU-1, BLEU-2, BLEU-3, BLEU-4, METEOR and ROUGE-L.

6.2 Future Work

The results achieved by our experiments demonstrate the effectiveness of our methods in the question generation problem. However, it would be interesting to explore the effect of various directions:

- Deriving the sentence embedding in the proposed concept-aware model using LSTMs instead of obtaining average over the word feature vectors.
- Utilizing conceptual word representation, such as BERT, ELMo, GPT-2 in the input layer so that word representation is based on its context.
- Demonstrate the effectiveness of our models on paragraph level question generation approaches.
- Investigate the impact of using other available lexical databases, such as ConceptNet (Speer, Chin, and Havasi, 2016) and Freebase (Bollacker et al., 2008) and incorporate the conceptual word representations defined in these knowledge bases into the NQG task.

References

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton (2016). *Layer Normalization*. arXiv: 1607.06450 [stat.ML].

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014). "Neural machine translation by jointly learning to align and translate". English (US). In: *arXiv*.

Banerjee, Satanjeev and Alon Lavie (June 2005). "METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments". In: *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*. Ann Arbor, Michigan: Association for Computational Linguistics, pp. 65–72. URL: https://www.aclweb.org/anthology/W05-0909.

Bengio, Y., P. Simard, and P. Frasconi (1994). "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Transactions on Neural Networks* 5.2, pp. 157–166.

Bengio, Yoshua et al. (Mar. 2003). "A Neural Probabilistic Language Model". In: *J. Mach. Learn. Res.* 3.null, pp. 1137–1155. ISSN: 1532-4435.

Bikel, Daniel M., Richard M. Schwartz, and Ralph M. Weischedel (1999). "An Algorithm that Learns What's in a Name". In: *Machine Learning* 34, pp. 211–231.

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://www.aclweb.org/anthology/P04-3031.

Blei, David M., Andrew Y. Ng, and Michael I. Jordan (2003). "Latent dirichlet allocation". In: *The Journal of Machine Learning Research* 3, pp. 993–1022. ISSN: 1532-4435.

Bojar, Ondřej, Kamil Kos, and David Mareček (July 2010). "Tackling Sparse Data Issue in Machine Translation Evaluation". In: *Proceedings of the ACL 2010 Conference Short Papers*. Uppsala, Sweden: Association for Computational Linguistics, pp. 86–91. URL: https://www.aclweb.org/anthology/P10-2016.

Bollacker, Kurt et al. (Jan. 2008). "Freebase: A collaboratively created graph database for structuring human knowledge". In: pp. 1247–1250. DOI: 10.1145/1376616.1376746.

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.

Chen, Guanliang et al. (2018). "LearningQ: a large-scale dataset for educational question generation". English. In: *Proceedings of the Twelfth International Conference on Web and Social Media*. Ed. by Kate Starbird and Ingmar Weber. International AAAI Conference on Weblogs and Social Media 2018, ICWSM 2018 ; Conference date: 25-06-2018 Through 28-06-2018. United States of America: Association for the Advancement of Artificial Intelligence (AAAI), pp. 481–490. URL: https://icwsm.org/2018/.

Chen, Stanley F. and Joshua Goodman (1996). "An Empirical Study of Smoothing Techniques for Language Modeling". In: *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*. ACL '96. Santa Cruz, California: Association for Computational Linguistics, pp. 310–318. DOI: 10.3115/981863.981904. URL: https://doi.org/10.3115/981863.981904.

Chen, Wei and Gregory Aist (2009). "Generating Questions Automatically from Informational Text". In:

Chiticariu, Laura et al. (2010). "Domain Adaptation of Rule-based Annotators for Named-entity Recognition Tasks". In: *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. EMNLP '10. Cambridge, Massachusetts: Association for Computational Linguistics, pp. 1002–1012. URL: http://dl.acm.org/citation.cfm?id=1870658.1870756.

Cho, KyungHyun et al. (2014). "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches". In: *CoRR* abs/1409.1259. arXiv: 1409.1259. URL: http://arxiv.org/abs/1409.1259.

Cho, Kyunghyun et al. (2014). "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *CoRR* abs/1406.1078. arXiv: 1406.1078. URL: http://arxiv.org/abs/1406.1078.

Chung, Yu-An, Hung-yi Lee, and James R. Glass (2017). "Supervised and Unsupervised Transfer Learning for Question Answering". In: *CoRR* abs/1711.05345. arXiv: 1711.05345. URL: http://arxiv.org/abs/1711.05345.

Collobert, Ronan et al. (2011). "Natural Language Processing (almost) from Scratch". In: *CoRR* abs/1103.0398. arXiv: 1103.0398. URL: http://arxiv.org/abs/1103.0398.

Conneau, Alexis et al. (2017). "Supervised Learning of Universal Sentence Representations from Natural Language Inference Data". In: *CoRR* abs/1705.02364. arXiv: 1705.02364. URL: http://arxiv.org/abs/1705.02364.

Devlin, Jacob et al. (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805. arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

Du, Xinya, Junru Shao, and Claire Cardie (2017). "Learning to Ask: Neural Question Generation for Reading Comprehension". In: *CoRR* abs/1705.00106. arXiv: 1705.00106. URL: http://arxiv.org/abs/1705.00106.

Duan, Nan et al. (Sept. 2017). "Question Generation for Question Answering". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 866–874. DOI: 10.18653/v1/D17-1090. URL: https://www.aclweb.org/anthology/D17-1090.

El-Kishky, Ahmed et al. (2014). "Scalable Topical Phrase Mining from Text Corpora". In: *CoRR* abs/1406.6312. arXiv: 1406.6312. URL: http://arxiv.org/abs/1406.6312.

Felbo, Bjarke et al. (2017). "Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. DOI: 10.18653/v1/d17-1169. URL: http://dx.doi.org/10.18653/v1/D17-1169.

Finkel, Jenny Rose, Trond Grenager, and Christopher Manning (2005). "Incorporating Non-Local Information into Information Extraction Systems by Gibbs

Sampling". In: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. ACL '05. Ann Arbor, Michigan: Association for Computational Linguistics, pp. 363–370. DOI: 10.3115/1219840.1219885. URL: https://doi.org/10.3115/1219840.1219885.

Gage, Philip (Feb. 1994). "A New Algorithm for Data Compression". In: *C Users J.* 12.2, pp. 23–38. ISSN: 0898-9788.

Gates, Donna (Jan. 2008). "Generating Look-Back Strategy Questions from Expository Texts". In:

Gu, Jiatao et al. (Aug. 2016). "Incorporating Copying Mechanism in Sequence-to-Sequence Learning". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 1631–1640. DOI: 10.18653/v1/P16-1154. URL: https://www.aclweb.org/anthology/P16-1154.

Gülçehre, Çaglar et al. (2016). "Pointing the Unknown Words". In: *CoRR* abs/1603.08148. arXiv: 1603.08148. URL: http://arxiv.org/abs/1603.08148.

Harrison, Vrindavan and Marilyn A. Walker (2018). "Neural Generation of Diverse Questions using Answer Focus, Contextual and Linguistic Features". In: *CoRR* abs/1809.02637. arXiv: 1809.02637. URL: http://arxiv.org/abs/1809.02637.

He, Kaiming et al. (2015). "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385. arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

Heilman, Michael (2011). "Automatic Factual Question Generation from Text". PhD thesis. USA. ISBN: 9781267582249.

Heilman, Michael and Noah A. Smith (June 2010). "Good Question! Statistical Ranking for Question Generation". In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Los Angeles, California: Association for Computational Linguistics, pp. 609–617. URL: https://www.aclweb.org/anthology/N10-1086.

Hochreiter, Sepp and Jürgen Schmidhuber (Dec. 1997). "Long Short-term Memory". In: *Neural computation* 9, pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

Howard, Jeremy and Sebastian Ruder (2018). "Fine-tuned Language Models for Text Classification". In: *CoRR* abs/1801.06146. arXiv: 1801.06146. URL: http://arxiv.org/abs/1801.06146.

Hu, Ronghang et al. (2015). "Natural Language Object Retrieval". In: *CoRR* abs/1511.04164. arXiv: 1511.04164. URL: http://arxiv.org/abs/1511.04164.

Huang, Yu-Hsiang (2017a). "Attention-is-all-you-need-pytorch". In: https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html.

— (2017b). "Attention-is-all-you-need-pytorch". In: https://github.com/jadore801120/attention-is-all-you-need-pytorch.

Karpathy, Andrej and Fei-Fei Li (2014). "Deep Visual-Semantic Alignments for Generating Image Descriptions". In: *CoRR* abs/1412.2306. arXiv: 1412.2306. URL: http://arxiv.org/abs/1412.2306.

Katz, S. (1987). "Estimation of probabilities from sparse data for the language model component of a speech recognizer". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 35.3, pp. 400–401.

Kim, Yanghoon et al. (2018). "Improving Neural Question Generation using Answer Separation". In: *CoRR* abs/1809.02393. arXiv: 1809.02393. URL: http://arxiv.org/abs/1809.02393.

Kiros, Ryan et al. (2015). "Skip-Thought Vectors". In: *CoRR* abs/1506.06726. arXiv: 1506.06726. URL: http://arxiv.org/abs/1506.06726.

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

Leidner, Jochen L. et al. (2003). "Automatic Multi-Layer Corpus Annotation for Evaluation Question Answering Methods: CBC4Kids". In: *Proceedings of 4th International Workshop on Linguistically Interpreted Corpora (LINC-03) at EACL 2003*. URL: https://www.aclweb.org/anthology/W03-2406.

Lin, Chin-Yew (July 2004). "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, pp. 74–81. URL: https://www.aclweb.org/anthology/W04-1013.

Ling, Wang et al. (2015). "Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation". In: *CoRR* abs/1508.02096. arXiv: 1508.02096. URL: http://arxiv.org/abs/1508.02096.

Liu, Fangxu et al. (2019). "Improving Performance of NMT Using Semantic Concept of WordNet Synset". In: *Machine Translation*. Ed. by Jiajun Chen and Jiajun Zhang. Singapore: Springer Singapore, pp. 39–51. ISBN: 978-981-13-3083-4.

Liu, Ming, Rafael A. Calvo, and Vasile Rus (2010). "Automatic Question Generation for Literature Review Writing Support". In: *Proceedings of the 10th International Conference on Intelligent Tutoring Systems - Volume Part I*. ITS'10. Pittsburgh, PA: Springer-Verlag, pp. 45–54. ISBN: 3642133878. DOI: 10.1007/978-3-642-13388-6_9. URL: https://doi.org/10.1007/978-3-642-13388-6_9.

Long, Mingsheng et al. (2015). "Learning Transferable Features with Deep Adaptation Networks". In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, pp. 97–105.

Luong, Minh-Thang and Christopher D. Manning (2016). "Achieving Open Vocabulary Neural Machine Translation with Hybrid Word-Character Models". In: *CoRR* abs/1604.00788. arXiv: 1604.00788. URL: http://arxiv.org/abs/1604.00788.

Luong, Thang, Hieu Pham, and Christopher D. Manning (Sept. 2015). "Effective Approaches to Attention-based Neural Machine Translation". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 1412–1421. DOI: 10.18653/v1/D15-1166. URL: https://www.aclweb.org/anthology/D15-1166.

Luong, Thang, Ilya Sutskever, et al. (2014). "Addressing the Rare Word Problem in Neural Machine Translation". In: *CoRR* abs/1410.8206. arXiv: 1410.8206. URL: http://arxiv.org/abs/1410.8206.

Manning, Christopher et al. (June 2014). "The Stanford CoreNLP Natural Language Processing Toolkit". In: *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Baltimore, Maryland: Association for Computational Linguistics, pp. 55–60. DOI: 10.3115/v1/P14-5010. URL: https://www.aclweb.org/anthology/P14-5010.

Mazidi, Karen and Rodney D. Nielsen (June 2014). "Linguistic Considerations in Automatic Question Generation". In: *Proceedings of the 52nd Annual Meeting of*

the Association for Computational Linguistics (Volume 2: Short Papers). Baltimore, Maryland: Association for Computational Linguistics, pp. 321–326. DOI: 10.3115/v1/P14-2053. URL: https://www.aclweb.org/anthology/P14-2053.

Mazidi, Karen and Rodney D. Nielsen (2015). "Leveraging Multiple Views of Text for Automatic Question Generation". In: *Artificial Intelligence in Education*. Ed. by Cristina Conati et al. Cham: Springer International Publishing, pp. 257–266. ISBN: 978-3-319-19773-9.

McCann, Bryan et al. (2017). "Learned in Translation: Contextualized Word Vectors". In: *CoRR* abs/1708.00107. arXiv: 1708.00107. URL: http://arxiv.org/abs/1708.00107.

Mi, Haitao et al. (Nov. 2016). "Coverage Embedding Models for Neural Machine Translation". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, pp. 955–960. DOI: 10.18653/v1/D16-1096. URL: https://www.aclweb.org/anthology/D16-1096.

Mikolov, Tomas, Kai Chen, et al. (2013). *Efficient Estimation of Word Representations in Vector Space*. URL: http://arxiv.org/abs/1301.3781.

Mikolov, Tomas, Ilya Sutskever, et al. (2013). "Distributed Representations of Words and Phrases and their Compositionality". In: *CoRR* abs/1310.4546. arXiv: 1310.4546. URL: http://arxiv.org/abs/1310.4546.

Miller, George A. et al. (Dec. 1990). "Introduction to WordNet: An On-line Lexical Database*". In: *International Journal of Lexicography* 3.4, pp. 235–244. ISSN: 0950-3846. DOI: 10.1093/ijl/3.4.235. eprint: https://academic.oup.com/ijl/article-pdf/3/4/235/9820417/235.pdf. URL: https://doi.org/10.1093/ijl/3.4.235.

Mitkov, Ruslan and Le An Ha (2003). "Computer-Aided Generation of Multiple-Choice Tests". In: *Proceedings of the HLT-NAACL 03 Workshop on Building Educational Applications Using Natural Language Processing*, pp. 17–22. URL: https://www.aclweb.org/anthology/W03-0203.

Mostafazadeh, Nasrin et al. (2016). "Generating Natural Questions About an Image". In: *CoRR* abs/1603.06059. arXiv: 1603.06059. URL: http://arxiv.org/abs/1603.06059.

Mou, Lili et al. (Nov. 2016). "How Transferable are Neural Networks in NLP Applications?" In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, pp. 479–489. DOI: 10.18653/v1/D16-1046. URL: https://www.aclweb.org/anthology/D16-1046.

Nguyen, Tri et al. (2016). "MS MARCO: A Human Generated MAchine Reading COmprehension Dataset". In: *CoRR* abs/1611.09268. arXiv: 1611.09268. URL: http://arxiv.org/abs/1611.09268.

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. DOI: 10.1162/0891201053630264. URL: https://www.aclweb.org/anthology/J05-1004.

Pan, Liangming et al. (2019). "Recent Advances in Neural Question Generation". In: *CoRR* abs/1905.08949. arXiv: 1905.08949. URL: http://arxiv.org/abs/1905.08949.

Pan, S. J. and Q. Yang (2010). "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10, pp. 1345–1359.

Papineni, Kishore et al. (2002). "BLEU: A Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. ACL '02. Philadelphia, Pennsylvania: Association for Computational Linguistics, pp. 311–318. DOI: 10.3115/1073083.1073135. URL: https://doi.org/10.3115/1073083.1073135.

Parikh, Ankur P. et al. (2016). "A Decomposable Attention Model for Natural Language Inference". In: *CoRR* abs/1606.01933. arXiv: 1606.01933. URL: http://arxiv.org/abs/1606.01933.

Paulus, Romain, Caiming Xiong, and Richard Socher (2017). "A Deep Reinforced Model for Abstractive Summarization". In: *CoRR* abs/1705.04304. arXiv: 1705.04304. URL: http://arxiv.org/abs/1705.04304.

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://www.aclweb.org/anthology/D14-1162.

Peters, Matthew E., Waleed Ammar, et al. (2017). "Semi-supervised sequence tagging with bidirectional language models". In: *CoRR* abs/1705.00108. arXiv: 1705.00108. URL: http://arxiv.org/abs/1705.00108.

Peters, Matthew E., Mark Neumann, et al. (2018). "Deep contextualized word representations". In: *CoRR* abs/1802.05365. arXiv: 1802.05365. URL: http://arxiv.org/abs/1802.05365.

Radford, Alec (2018). "Improving Language Understanding by Generative Pre-Training". In:

Rajpurkar, Pranav et al. (2016). "SQuAD: 100, 000+ Questions for Machine Comprehension of Text". In: *CoRR* abs/1606.05250. arXiv: 1606.05250. URL: http://arxiv.org/abs/1606.05250.

Reddy, Sathish et al. (Apr. 2017). "Generating Natural Language Question-Answer Pairs from a Knowledge Graph Using a RNN Based Question Generation Model". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, pp. 376–385. URL: https://www.aclweb.org/anthology/E17-1036.

Richardson, Matthew, Christopher J.C. Burges, and Erin Renshaw (Oct. 2013). "MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, pp. 193–203. URL: https://www.aclweb.org/anthology/D13-1020.

Rokhlenko, Oleg and Idan Szpektor (Aug. 2013). "Generating Synthetic Comparable Questions for News Articles". In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 742–751. URL: https://www.aclweb.org/anthology/P13-1073.

Rothe, Anselm, Brenden M. Lake, and Todd M. Gureckis (2017). "Question Asking as Program Generation". In: *CoRR* abs/1711.06351. arXiv: 1711.06351. URL: http://arxiv.org/abs/1711.06351.

Ruder, Sebastian (2017). "Transfer Learning - Machine Learning's Next Frontier". In: https://ruder.io/transfer-learning/.

Rush, Alexander M., Sumit Chopra, and Jason Weston (Sept. 2015). "A Neural Attention Model for Abstractive Sentence Summarization". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 379–389. DOI: 10.18653/v1/D15-1044. URL: https://www.aclweb.org/anthology/D15-1044.

Russakovsky, Olga et al. (2014). "ImageNet Large Scale Visual Recognition Challenge". In: *CoRR* abs/1409.0575. arXiv: 1409.0575. URL: http://arxiv.org/abs/1409.0575.

Sankaran, Baskaran et al. (2016). *Temporal Attention Model for Neural Machine Translation*. arXiv: 1608.02927 [cs.CL].

Schuster, M. and K. K. Paliwal (1997). "Bidirectional recurrent neural networks". In: *IEEE Transactions on Signal Processing* 45.11, pp. 2673–2681.

See, Abigail, Peter J. Liu, and Christopher D. Manning (2017). "Get To The Point: Summarization with Pointer-Generator Networks". In: *CoRR* abs/1704.04368. arXiv: 1704.04368. URL: http://arxiv.org/abs/1704.04368.

Sennrich, Rico, Barry Haddow, and Alexandra Birch (2015). "Neural Machine Translation of Rare Words with Subword Units". In: *CoRR* abs/1508.07909. arXiv: 1508.07909. URL: http://arxiv.org/abs/1508.07909.

Sharma, Shikhar et al. (2017). "Relevance of Unsupervised Metrics in Task-Oriented Dialogue for Evaluating Natural Language Generation". In: *CoRR* abs/1706.09799. URL: http://arxiv.org/abs/1706.09799.

Speer, Robyn, Joshua Chin, and Catherine Havasi (2016). "ConceptNet 5.5: An Open Multilingual Graph of General Knowledge". In: *CoRR* abs/1612.03975. arXiv: 1612.03975. URL: http://arxiv.org/abs/1612.03975.

Sun, Xingwu et al. (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://www.aclweb.org/anthology/D18-1427.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le (2014). "Sequence to Sequence Learning with Neural Networks". In: *CoRR* abs/1409.3215. arXiv: 1409.3215. URL: http://arxiv.org/abs/1409.3215.

Tapaswi, Makarand et al. (2015). "MovieQA: Understanding Stories in Movies through Question-Answering". In: *CoRR* abs/1512.02902. arXiv: 1512.02902. URL: http://arxiv.org/abs/1512.02902.

Tseng, Bo-Hsiang et al. (2016). *Towards Machine Comprehension of Spoken Content: Initial TOEFL Listening Comprehension Test by Machine*. arXiv: 1608.06378 [cs.CL].

Vaswani, Ashish et al. (2017). "Attention Is All You Need". In: *CoRR* abs/1706.03762. arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

Wang, D. and T. F. Zheng (2015). "Transfer learning for speech and language processing". In: *2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pp. 1225–1237.

Wang, Wenhui et al. (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.

Wolfe, John H. (Feb. 1976). "Automatic Question Generation from Text - an Aid to Independent Study". In: *SIGCUE Outlook* 10.SI, pp. 104–112. ISSN: 0163-5735. DOI: 10.1145/953026.803459. URL: https://doi.org/10.1145/953026.803459.

Xiong, Caiming, Victor Zhong, and Richard Socher (2016). "Dynamic Coattention Networks For Question Answering". In: *CoRR* abs/1611.01604. arXiv: 1611.01604. URL: http://arxiv.org/abs/1611.01604.

Yang, Zichao et al. (2015). "Stacked Attention Networks for Image Question Answering". In: *CoRR* abs/1511.02274. arXiv: 1511.02274. URL: http://arxiv.org/abs/1511.02274.

Yao, Xuchen, Gosse Bouma, and Yi Zhang (2012). "Semantics-based Question Generation and Implementation". In: *DD* 3, pp. 11–42.

Zhao, Yao et al. (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://www.aclweb.org/anthology/D18-1424.

Zhou, Qingyu et al. (2017). "Neural Question Generation from Text: A Preliminary Study". In: *CoRR* abs/1704.01792. arXiv: 1704.01792. URL: http://arxiv.org/abs/1704.01792.

Zhu, Yukun et al. (2015). "Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books". In: *CoRR* abs/1506.06724. arXiv: 1506.06724. URL: http://arxiv.org/abs/1506.06724.

Appendices

APPENDIX

Appendix One

Supplementary Material

This appendix provides list of top words extracted from our target domain (i.e., car manuals) using the LDA topic modeling:

- switch
- position
- ignition
- engine
- button
- start
- stop
- push
- key
- turn
- Odometer
- fob
- lever
- gear
- enter-n-go
- addition
- acc
- installing
- keyless
- pushed
- return
- wheel
- indicator
- active
- driven
- tcs
- activation
- malfunction
- wheels
- spin
- partial
- feature
- acceleration
- limited
- throttle
- spinning

- sway
- chime
- brake
- park
- remains
- minimum
- pedal
- shift
- torque
- sound
- anti-lock
- transmission
- starts
- sensors
- braking
- gear
- pressure
- road
- assist
- cluster
- low
- emergency
- applied
- lever
- service
- power
- abs
- drive
- monitoring
- automatically
- parking
- range
- evic
- dashes
- brakes
- neutral
- tpm
- identified
- bas
- econ
- tpms
- monitor
- foot
- interlock
- compact
- abnormal
- hill
- selector
- place
- components
- short
- reverse
- telltale
- present
- amount
- miles
- fault
- additives
- function
- instrument
- graphic
- content
- fully
- shifts
- inflate
- stuck
- avoid
- gears

- improve
- gasoline
- performance
- provide
- mmt
- reformulated
- manufacturer
- reduce
- blended
- octane
- emissions
- quality
- spark
- fall
- motor
- light
- warning
- located
- required
- driving
- turned
- oils
- repair
- functioning
- ease
- adjust
- interference
- comfort
- important
- counteracting
- sequence
- serviced
- monitors
- conditions
- steering
- control
- reduced
- stability
- ride
- increase
- intended
- path
- maintain
- functions
- power
- designation
- slow
- stopping
- sand
- understeer
- oversteer
- over/under
- tires
- tread
- wear
- snow
- safety
- replacement
- season
- life
- sidewall
- summer
- handling
- ice
- affect
- symbol
- adversely
- failure

- winter
- maintenance
- sets
- engine
- normal
- release
- accelerator
- pedal
- heater
- block
- press
- disengage
- starter
- hold
- weather
- flooded
- fails
- temperatures
- cord
- integrated
- front
- side
- rear
- label
- axle
- rating
- weight
- driver's
- door
- tin
- gawr
- full
- found
- capacity
- gross
- gvwr
- exceed
- certification
- states
- higher
- tire
- pressure
- inflation
- cold
- recommended
- loading
- pressures
- placard
- maximum
- b-pillar
- code
- check
- high
- audible
- levels
- kpa
- description
- activated
- speeds
- speed
- operation
- economy
- km/h
- mph
- traction
- section
- operate
- equivalent

- driving
- towing
- capability
- safe
- increased
- standing
- indicators
- powered
- lower
- required
- replace
- hesitations
- mounted
- ffv
- service
- referred
- make
- deterioration
- temporary
- uphill
- additional
- underinflated
- replaced
- met
- reasons
- results
- limited-use
- sufficient
- detected
- shallow
- reinstall
- fuels
- power
- decrease
- worn
- world-wide
- standard
- idle
- type
- update
- fuel
- liter
- match
- premium
- ethanol
- mopar
- firmly
- designs
- unleaded
- tire
- precautions
- electronic
- flexible
- spare
- load
- cap
- properly
- size
- water
- instrument
- compatible
- original
- designed
- center
- refueling
- equipment
- damage
- filler

- panel
- warnings
- surfaces
- left
- install
- actual
- evic
- loose
- gas
- run
- traction
- rotation
- trailer
- stop
- flat
- subsection
- driving
- device
- proper
- tsc
- wet
- swaying
- loss
- rapid
- limited
- pattern
- patterns
- hsa
- driver
- operating
- activation
- greater
- running
- improper
- criteria
- malfunction
- slippery
- module
- seat
- non-flex
- oxygenates
- receiver
- limits
- consumption
- molded
- activate
- tire
- pressure
- spare
- tire
- ignition
- switch
- shift
- lever
- Tire
- Pressure
- Monitoring
- TelltaleLight
- lbs
- kg
- brake
- pedal
- ENGINE
- STARt
- front
- rear
- tiresize
- parking

- brake
- original
- equipment
- Indicator
- Light
- cold
- tire
- inflation
- pressure
- mph
- km
- trailer
- towing
- apply
- parking
- brake
- towing
- trailer
- shift
- transmission
- Transmission
- Shift
- inflation
- pressure
- cold
- tire
- compact
- spare
- tire
- inflation
- pressure
- parking
- brake
- applied
- inflation
- pressure
- Inflate
- Monitoring
- spare
- pedal
- electronic
- distance
- reduce
- position
- emissions
- covered
- ahead
- reformulated
- parts
- inflated
- stationary
- season
- loading
- exploited
- children
- key
- panel
- complete
- wheel
- spare
- limited
- equipped
- compact
- temporary
- emergency
- valve
- designed
- filling

- receive
- spin
- monitor
- mounted
- require
- traction
- aftermarket
- travel
- tire
- pressure
- inflation
- cold
- low
- monitoring
- telltale
- placard
- kpa
- psi
- increased
- limit
- hours
- update
- activated
- recommended
- module
- updated
- extinguish
- control
- result
- cruise
- adaptive
- loss
- resulting
- injury
- normal
- lose
- gear
- detect
- install
- death
- lbs
- pump
- repaired
- foot
- electronic
- situations
- front
- rear
- wheels
- safety
- maintain
- axle
- important
- installed
- short
- shift
- area
- user
- bas
- range
- ready
- regularly
- tire
- tpms
- pressures
- original
- equipment
- sensors

- condition
- replace
- center
- stem
- reinstall
- devices
- ride
- improper
- sidewall
- molded
- contact
- ratings
- trailers
- gasoline
- driver
- side
- door
- located
- components
- ethanol
- label
- pillar
- occur
- axles
- requirements
- warranty
- unleaded
- handling
- listed
- symbol
- surfaces
- function
- fuel
- failure
- cap
- gap
- provide
- filler
- performance
- left
- tank
- longer
- roads
- gas
- unequal
- blended
- quality
- closed
- passengers
- displays
- gauge
- weight
- load
- maximum
- exceed
- loading
- cargo
- gawr
- capacity
- loaded
- gvwr
- total
- occupants
- certification
- combined
- carrying
- distribute
- winter

- kg
- tires
- tread
- snow
- wear
- run
- life
- ice
- select
- replaced
- handling
- indicators
- summer
- part
- sidewall
- designation
- higher
- improve
- repair
- damage
- steering
- size
- power
- required
- transmission
- operation
- starting
- section
- increase
- fluid
- rating
- acceleration
- chains
- performance
- occurs
- normal
- evenly
- replacing
- trailer
- brakes
- towing
- tow
- hitch
- follow
- tongue
- sway
- dangerous
- avoid
- heavier
- stop
- found
- swaying
- manual
- appears
- ambient
- harness
- device
- braking
- lock
- anti
- manufacturer
- replacement
- abs
- order
- lead
- move
- compatible
- setting

- scale
- hydraulic
- noise
- chart
- completely
- drivetrain
- driving
- conditions
- road
- traction
- drive
- water
- standing
- operate
- stopping
- rotation
- additional
- lane
- path
- due
- lower
- correct
- miles
- limits
- engine
- switch
- ignition
- temperature
- turn
- high
- check
- turned
- start
- automatically
- heavy
- running
- grade
- period
- octane
- damaged
- stopped
- immediately
- recommended
- refer
- proper
- make
- speeds
- stop
- operating
- maintenance
- affect
- checked
- safe
- full
- sensor
- properly
- capability
- continue
- tpms
- brake
- parking
- warning
- collision
- apply
- assist
- park
- applied
- hill

- hsa
- tsc
- fully
- slow
- automatic
- lamps
- received
- sounds
- interference
- speed
- km
- set
- mph
- pedal
- driven
- press
- release
- approximately
- minutes
- change
- accelerator
- device
- greater
- res
- loads
- turns
- index
- hydroplaning
- esc
- indicator
- malfunction
- prevent
- stability
- tcs
- level
- flat
- partial
- reduced
- including
- excessive
- feature
- laws
- condition
- disabled
- momentarily
- light
- display
- service
- message
- seconds
- fault
- active
- warning
- chime
- evic
- activation
- sound
- graphic
- remain
- remains
- instrument
- place
- tpm