Interview Question Generation [*]

Deep Inder Mohan

October 2019

1 Introduction

This is a report of the work done in summer, 2019, as a part of a research internship under Prof. Dinesh Babu Jayagopi[1] and in the direct supervision of my mentor Ms. Pooja Rao[2].

1.1 Motivation

Many tasks or problems have been automated or solved with the use machine learning and neural networks. The aim of a neural network is generally to reduce to human effort and to solve the problem efficiently. Interview systems, with growing number of candidates, are adopting to asynchronous video interview format and online assessment of candidates. These systems pose questions in a predefined order or generally select a question randomly from a pool of questions in contrast to traditional conversational interview. This makes the entire process very unnatural as there is often little to know overlap between the answer to a question and the next question asked. This can be addressed by taking the candidate's response into account and asking the right follow up question.

1.2 Problem Statement

Automatic question generation (QG) is a challenging problem in NLP. QG systems are typically built assuming access to a large number of training instances where each instance is a passage and corresponding questions generated on that passage. Example:-

[*] https://github.com/deep-guy/interviewQG
[1] jdinesh@iiitb.ac.in
[2] pooja.rao@iiitb.org

Text passage: ...the court of justice accepted that a requirement to speak gaelic to teach in a dublin design college could be justified as part of the public policy of promoting the irish language.
Generated Questions: what did the court of justice not claim to do?; what language did the court of justice decide to speak?

The Interview Question Generation problem, which is was the problem statement for this project, is similar to the Automatic Question Generation task, and has the following definition:-

"Given a question asked in an HR interview, and the corresponding answer to that question, generate a relevant followup question that the interviewer can ask the interviewee."

While the Automatic Question Generation problem focuses on generating questions whose answers are contained within the input text passage, Interview Question Generation involves generating questions using contextual cues provided by the input text, but not with respect to some answer phrases in the input.

Question: Give an example of how you worked in team.
Answer: I once worked in a team of 5 individuals. 2 of them (including myself) were interested in working on the technicalities of the project while others were not. Hence, we divided the project work among ourselves depending on our interests. 2 of us worked on the technicalities of the project, one on implementation of those technicalities and while rest two of them helped in documentation of the project.
Relevant follow-up question: What was the focus of the project ?

To achieve a viable solution of the problem statement mentioned above, I segmented it into the following sub problems:-

1. How would I **extract** 'relevance' as a parameter from my given <question-answer> pair?

2. Given some extracted 'relevant' data, how would I use it to **generate** questions?

3. How would I **judge** the accuracy of a generated question in terms of its relevance?

2 Literature Survey

2.1 Background

A sequence to sequence model(seq2seq model) also known as encoder-decoder model as it consist of an encoder module which take the input sequence and encodes it and another decoder module which takes the encoded sequence and decodes it into output. Both the encoder and decoder are individual RNNs.

A seq2seq model is built using several recurrent units. This is a Recurrent Neural Network. RNNs are of many types. The basic or vanilla RNN is not used for complex tasks. It can be used for sequences with small length since it has a vanishing gradient problem. Generally, special kinds of RNNs are used. For example, GRU, LSTM. Another technique used is reading the sequence in both directions, for example Bi-LSTM. To enhance the performance and address some specific issues other mechanisms like attention[1] mechanism and copy[2] mechanism are used.

2.1.1 Recurrent Neural Network

Recurrent neural network is a class of neural networks. What makes a recurrent neural network unique is its ability to take variable sized input. Illustrated in 1.

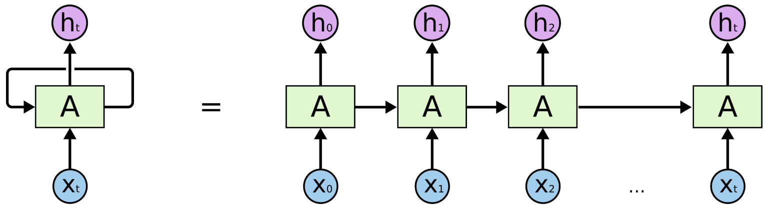


Figure 1: RNN. Image credits to (Chris Olah)[3]

This is possible as, unlike a feed forward network which relies only on present input, a RNN takes not just the present input but also the state information until the last input(feedback). This gives the RNN a sort of memory or ability to work with context. This makes it an effective model for sequential tasks. But the simple or vanilla RNN has a major drawback of vanishing gradient and can be used only for short sequences. We use Long Short Term Memory(LSTM) [4] or Gated Recurrent Units(GRU) for longer input sequences.

LSTM is a type of RNN which can remember long term dependencies. For this purpose it uses something called a cell state in each recurrent unit. The information in each of the cell state is controlled by three gates. The input gate, output gate and forget gate. Illustrated in 2

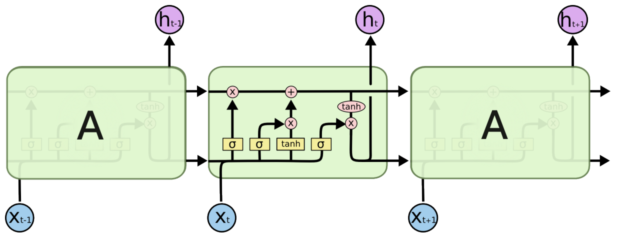


Figure 2: LSTM. Image credits to (Chris Olah)[3]

One of the variations of LSTM is GRU. In a GRU the input gate and forget gate are combined as update gate, rather than having two gates for what to input and what to forget, it has one gate for how to update to cell state. Attention[1] is generally used at the decoder step to specify which words to attend to in the input sequence (or in the earlier decoded sequence, self attention) to produce the output. Illustrated in 3 for a translation task.

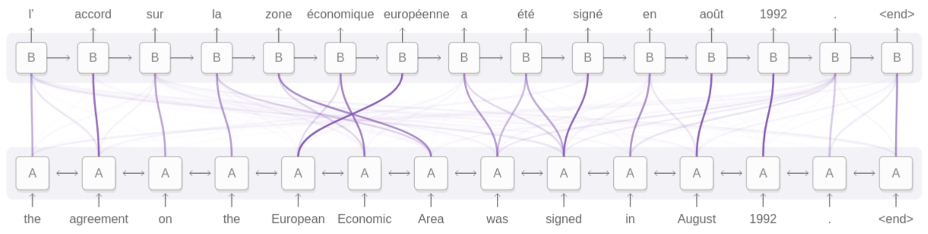


Figure 3: Attention. Image credits to (Chris Olah et al.)[5]

Another mechanism used is copy[2] mechanism. When the input sequence contains some span of text, like entity name, which is to be directly 'copied' to the output sequence we use copy mechanism.

2.2 Text Generation(Natural Language Generation)

Text generation is the task of generating text which is indistinguishable from human written text. Generally either statistical models are used for text generation which maximises the probability of target sentence given source sentence or rule based sentences are generated where the text is generated as per handcrafted rules by language experts. Although the rule based model which rely on template are more interpretable and understandable the statistical models are easily trainable and do not require an expert to handcraft the rules.

The generation can also be seen as text to text generation or data to text. Text to text is language generation when the input is linguistic where as data to text refers to text generation on non linguistic input like images, tables, networks, graphs and so on. In our case we require a text to text model. But along with the text we planned to provide some knowledge, in the form of triples or relations and other information like the traits which we decided to concatenate with the source. To provide knowledge we referred to some of the existing knowledge bases, one such knowledge base is ConceptNet. ConceptNet is an open knowledge graph which represents common words and phrases and relationship between them. When a knowledge base in used to provide input to the model it is called *Knowledge Based Text Generation.*

Since our model must take the context and generate follow-up question directly we prefer end to end systems. Also, end to end generally perform better in language generation tasks. We look into some the state of the art and other results from the End to End(E2E) text generation challenge. We also look into other text generation models.

2.2.1 Findings of E2E Challenge[6]

This paper summarises the experimental setup and results of the end to end natural language generation task in a spoken dialogue system. We observe that the paper shares the performance of several models submitted as a part of the challenge. In the paper the results of the models are compared on automatic evaluation as well as human evaluation. Major architectures used range from seq2seq based models as well as simple template based models to rule based model. Although the rule based model did not score as well as seq2seq in automatic scoring, upon human evaluation the naturalness of text generated by model of both the architectures was comparable. Hence showing that seq2seq architectures are data driven and very capable but require reranking to reach high quality.

2.2.2 A General Model for Neural Text Generation from Structured Data[7]

This paper introduces a end to end neural network model for data to text which aims to generate natural language description from structured data.

It uses a seq2seq model with copy mechanism and attention. This paper uses a interesting mechanism to discourage model from generating repetitive content. This is done using a coverage mechanism which keeps track of the words to which attention is giving when the generator decodes. In the study by Ziang Xie [8], repeated generation shows multiple attention to same word in the sequence. Illustrated in Figure 4.

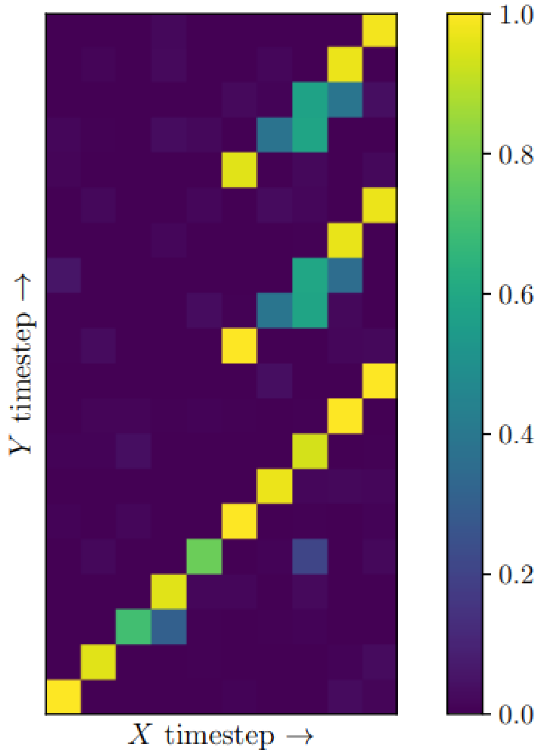


Figure 4: Attention for a repeating sequence. Image credits to Ziang Xie (2017)[8]

2.2.3 Deep Graph Convolutional Encoder for Structured Data to Text Generation[9]

This paper discusses use of Graph Convolution Network(GCN) as decoder for data to text generation from structured graph data. Other technique which is generally used is to linearise the input data and use standard seq2seq model. The paper compare both these approaches on two knowledge bases tasks namely WebNLG and SR11Deep generation task. The sequence encoded by GCN is decoded using LSTM with soft attention mechanism. The comparison shows that the text generated from GCN encoded sequence was less prone to under-generation or over-generation.

2.2.4 Learning Neural Templates for Text Generation [10]

The neural, encoder-decoder model have empirical advantage and are trainable. However they are largely uninterpretable and the output cannot be easily controlled. When text is generated by filling templates, it is interpretable and easily controllable.

2.3 Question Generation

Question generation is a sub task of language generation where the output is conditioned to be a question. More in specific factual question generation refers to task of generating questions that can be answered by certain subspans of the given input. Majorly two approach used as in case of text generation are neural text generation and rule based or template based text generation. Similarly even question generation can be traditional rule based or data driven seq2seq neural question generation. Question generation can be useful in many domains such as education, enquiry, conversation and in interview follow-up question generation in our case.

2.3.1 Learning to Ask: Neural Question Generation for Reading Comprehension[11]

This paper introduces a sequence to sequence model with attention mechanism to generate question for reading comprehension. It does not use a rule based approach rather builds an end to end data driven and trainable model. They compare their model in two ways. One is by varying the input information between sentence level and paragraph level information. Another by varying use of pre-trained vs learned word embeddings. They use the SQuAD dataset. The result show that when sentence level information is used over paragraph level the performance improves and also use pre-trained word embeddings out performs learning the word embeddings from scratch.

Brief model details -Conditional probability using RNN encoder-decoder architecture with attention. Bi-directional LSTM to encode the sentence as well as paragraph and to decode the encoded vector. They use LSTM with hidden unit size 600 and 2 layer LSTMs in both the encoder and the decoder.

2.3.2 Neural Question Generation from Text: A Preliminary Study[12]

This paper is a preliminary study of natural language question generation, factual question in specific. It uses a neural encoder-decoder model to generate factoid questions. It show that use of Parts of speech and Named entity recognition tags improves performance. To point to the answer words they use BIO tags. In this scheme, tag B denotes the start of an answer, tag I continues the answer and tag O marks words that do not form part of an answer.

They compare different systems starting the comparison with rule based system. They use seq2seq model with attention as baseline. They extend it in steps by including lexical features, copy mechanism, pretrained word embedding (GloVe [13]) one by one. And compare the performance of each model using BLEU evaluation.

2.3.3 Improving Neural Question Generation using Answer Separation [14]

This paper introduces a novel model to generate factual question for education material with answer separation. It shows that an encoder-decoder model tries to maximize the information in the output and therefore ends up including the answer token as part of the question generated. Although previous works have used special tags (such as BIO tags) to encounter this problem it is not very effective. The use of copy mechanism although increases the evaluation score it worsens this problem. Hence the mask the the answer token in the input they feed to the encoder. They propose a novel module called keyword-net to regain the information lost by masking the answer token. They use this module to encode the information about the answer token. The input to the decoder is the concatenation of this encoding and the encoded sequence. This model solves the problem of answer token being generated as part of question to a large extent and outperforms the state of the art model.

2.3.4 QG -Net: data-driven QG model for educational content[15]

This paper introduces a model to generate question specifically for educational content. Two major challenges faced by this paper include, one generation of fluent and relevant question and two limited training data specific to educational application. They make use of massive publicly available general purpose dataset to address both fluency and relevance, and limited training data. They too use a sequence to sequence model which sequentially takes the input and gives the output word by word.

2.3.5 Zero-Shot Question Generation from Knowledge Graphs for Unseen Predicates and Entity Types[16]

This paper introduces a model to generate questions for knowledge base triples ie: Subject, Object and Predicate in zero shot setting. It is trained on the knowledge triples from freebase knowledge base.

2.4 Existing Implementations of Encoder Decoders

Seq2Seq model takes as input a sequence of words(sentence or sentences) and generates an output sequence of words. It does so by use of the recurrent neural network (RNN). Although the vanilla version of RNN is rarely used, its more advanced version i.e. LSTM or GRU are used. This is because RNN suffers from the problem of vanishing gradient.
It mainly has two components i.e encoder and decoder.

2.4.1 Fastai[17]

It is a pytorch library which simplifies training fast an accurate models. The library is based on research into deep learning best practices undertaken at fast.ai It also provides with tutorials to learn and get started.

2.4.2 IBM/pytorch-seq2seq[18]

This is a complete suite for training sequence-to-sequence models[19] in PyTorch. It consists of several models and code to both train and infer using them.

2.4.3 eladhoffer/seq2seq.pytorch[20]

An open source framework for seq2seq models in PyTorch. Provides with independent modules to write reusable code.

2.4.4 allenNLP/Encoder-Decoder[21]

An open source NLP research library, built on PyTorch, for developing state-of-the-art deep learning models on a wide variety of linguistic tasks.

2.4.5 OpenNMT-py[22]

OpenNMT-py: Open-Source Neural Machine Translation is a Pytorch port of OpenNMT, an open-source (MIT) neural machine translation system. It is designed to be research friendly to try out new ideas in translation, summary, and many other domains.
It provides with translator which essentially is an Encoder-Decoder[23].

2.4.6 QG-Net[24]

QG-Net is a data-driven question generation model generating questions on educational content. The project is built on top of OpenNMT-py. Dataset used for training is SQuAD 1.0[25].

3 Interview Question Generation

3.1 Extracting Relevance: Keyword Extraction

I decided on keyword extraction as the method for deciding which parts of my input data were relevant. Keyword extraction focuses on extracting a set of certain words/phrases from some input text that can be considered to an accurate representation of the information contained in that text. The reason I chose this method for extracting relevance was because the extracted keywords are the terms giving context to my input. Hence, extracted keywords would be able to provide the context I need for generating the output question.

3.1.1 AllenNLP

I used AllenNLP, which is an open source library of NLP models, as my first approach to the Keyword Extraction task. Since I was coming from a place of having no knowledge of Machine Learning, this task presented me with the opportunity to learn things like data preprocessing. I followed AllenNLP's tutorial on how to get started with training and evaluating models using their library. I used their parts-of-speech tagger (which consists of a word embedding layer followed by an LSTM), but instead of training it on Brown Corpus, I trained it on Essential Terms dataset, which has every word in a sentence labelled on a scale of 0 to 5, based on how important that word is. The script for preprocessing the dataset can be found here.
The results on testing the model are:-

```
        accuracy = 0.6577
        accuracy3 = 0.8786
        loss = 0.8018
```

An example output generated by this model on some text from our dataset is:-

Input: *"I am a risk taker. After I failed to clear JEE first time, though I had some other good college option, I risked taking a year drop."*
Output: "I/0 am/0 a/0 risk/1 taker/1 ./0 After/0 I/1 failed/2 to/0 clear/1 JEE/1 first/1 time/2 ,/0 though/2 I/0 had/0 some/0 other/0 good/2 college/1 option/1 ,/0 I/0 risked/2 taking/4 a/0 year/0 drop/3 ./0"
Keywords: failed, time, though, good, risked, taking, drop

3.1.2 Rapid Automatic Keyword Extractor (RAKE)

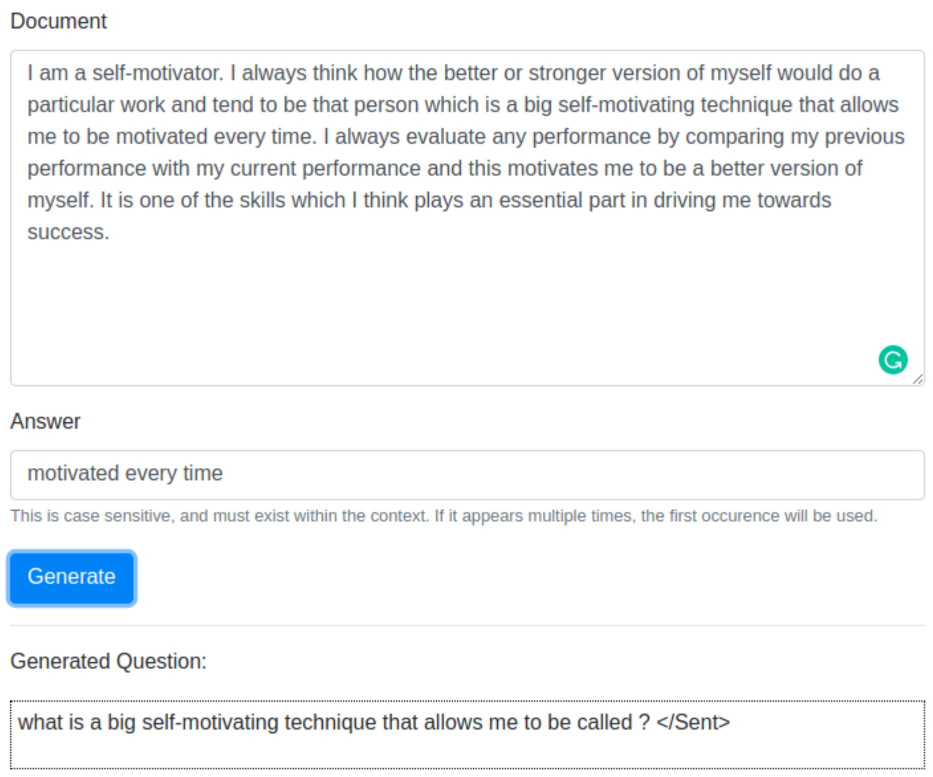
In the above model, since the training dataset is not related to our question-answer pairs, the results are relatively lackluster. Thus, I looked into other methods for keyword extraction, and I found RAKE, which is a domain independent keyword extraction algorithm. This algorithm works by first removing delimiters and stop words from in input data and identifying candidate phrases. It then builds a co-occurrence graph of those phrases and assigns a score to each phrase from the graph. The top scoring phrases are extracted. I used rake-nltk, which is a python implementation of this algorithm. The script for extracting keywords from our dataset, along with the raw dataset can be found here.
An example output generated from our dataset is:-

Input: *"I am a risk taker. After I failed to clear JEE first time, though I hadsome other good college option, I risked taking a year drop."*
Best extracted phrases: 'clear jee first time', 'good college option', 'year drop', 'risked taking', 'risk taker'

3.2 Question Generation Models

3.2.1 QG from passage

Having extracted keywords from the input data, the logical next step was to try to find a way to use these extracted keywords for question generation. I found a paper on Text-to-Text Neural Question Generation [26] where the author describes a model to generate questions on a given passage. This is an encoder-decoder QG model, which uses a bidirectional Long Short Term


Document

I am a self-motivator. I always think how the better or stronger version of myself would do a particular work and tend to be that person which is a big self-motivating technique that allows me to be motivated every time. I always evaluate any performance by comparing my previous performance with my current performance and this motivates me to be a better version of myself. It is one of the skills which I think plays an essential part in driving me towards success.

Answer

motivated every time

This is case sensitive, and must exist within the context. If it appears multiple times, the first occurence will be used.

Generated Question:

what is a big self-motivating technique that allows me to be called ? </Sent>

Figure 5: Sample result from Text-to-text Neural QG model

Memory (biLSTM) network in the encoder, and pointer-softmax copy mechanism in the decoder. I found an implementation of the model described in this paper, that also takes as input the answer phrase, and tries to generate a question from the passage with respect to that phrase [27]. The idea was to use the extracted phrases from RAKE and try to generate questions on them. The questions generated by the model, however, were neither valid followup questions, nor grammatically accurate. Figure 5 shows a sample result generated by this model.

This is primarily because (a) The model only tries to generate questions that can be answered from within the passage itself, and (b) Yuan et al.[26] trained their own Language Model (LM) on SQuAD questions, hence limiting the model's grammar capabilities in terms of generating interview questions (eg., interview questions usually have pronouns, whereas the questions in SQuAD contain primarily proper nouns). The code for this model can be found here.

3.2.2 OpenNMT + ConceptNet

While looking for other QG models, I found QG-net [**?**], which is an RNN-based model for generating questions from passages. QG-Net is an encoder-decoder question generation model. The encoder consists of two hierarchical RNN layers, and the decoder implements a hierarchical attention mechanism. Due to certain dependency issues, I was unable to replicate the results from their paper. But in their implementation, they were using OpenNMT [22], which is an open-source library of implementations of various translation and sequence learning models. The idea was to use their sequence to sequence 2 layer encoder-decoder translation RNN model, but instead of having English language sentences as source, and their translations as targets, I would have our interview answers as source, and their followup questions as the target.
The primary task in this approach was making the model learn to generate questions. The following are the approaches to this task:-

I. **SQuAD**
Since up to this point, all QG models I looked at were being trained on SQuAD, I thought it would be a good idea to train this model on SQuAD as well. This turned out to be the worst model, as all the questions being generated were from contexts that were only in SQuAD. So while they were grammatically sound, they had no relevance to the input data. Example:-

Answer (Input): "During my graduation, we have done a project on Face recognition. I was the leader ... greatest leadership achievement in a professional environment."
Followup (Output): who was schwarzenegger 's rival ?

The results can be found here.

II. **Followup Data**
My mentor had provided me with a dataset of around 20000 question-answer-followup triples to annotate. Since all the question answer pairs in this dataset would conform to the real world interview question-answers, I thought that training the model on this dataset might yield better results. The results, unlike with SQuAD, did not have any

random contexts, but they were largely incoherent questions, with certain parts-of-speech like articles and prepositions in incorrect places. Example:-

Answer (Input): I have absolutely no problem working under a person younger than me till ... learn new stuff and gather better experience from hisher work.
Followup (Output): Are you a good the life?

The results can be found here.

III. **RAKEd Data**
On looking at the results generated by the model above, it was clear to see that the model was giving too much importance to articles, prepositions, etc. My idea for circumventing this problem was to train the model on extracted keywords from answers, rather than the answers themselves, and also extract keywords before feeding them into the model during translation. Example:-

Answer: "During my graduation, we have done a project on Face recognition. I was the leader of my team ... that is the greatest leadership achievement in a professional environment."
Extracted Keywords (Input): greatest leadership achievement, working together, ... , project leader.
Followup (Output): Do you agree with me?

The results can be found here. This model generated this questions as this was one of the followups in the original dataset to this question answer pair. So this model is overfitting on the data.

IV. **ConceptNET**
My mentor had pointed out that the use of word embeddings can greatly improve the results of generative models. So I used ConceptNET [28] embeddings while preprocessing the data (extracted keywords) and trained a model. The performance of this model is objectively the best among the four. Example:-

Answer: "During my graduation, we have done a project on Face recognition. I was the leader of my team. In the peak stage of my project two of my teammates had a conflict ... greatest leadership achievement in a professional environment.
Extracted Keywords (Input): greatest leadership achievement, working together, ... , project leader.

Followup (Output): Are you sure that is the best way to solve the conflict?

The results can be found here

On further examining the results of the last two models, I realised that they were overfitting on the training data. This was because the data contains a limited number of <question-answer> pairs, but multiple followup questions corresponding to each pair (as many as 40 in some cases). Also, the followups are not natural questions, but artificially generated by my mentor's model (which itself, is not perfect). So, many of the followup questions in the training data itself are non-relevant. For example, the question generated by the keyword extractor model is one of the questions that her model generates.
This model would be able to generate much better results if the dataset consisted of distinct triples, with each followup question being relevant to the question and answer.

3.3 Evaluating questions: BERT Classifier

The next task was to create a model that could classify the questions generated by the QG models as 'good' or 'bad' on some scale. My mentor provided me with a dataset of about 900 <question-answer-followup> triples, with each followup question having a rating for grammar and relevance on a scale of 1 to 3.
Some research on the text classification problem revealed that a good way to go about it was to fine-tune some pretrained Language Model. I decided too use Google AI's BERT [29] Language Model(LM), as all sources indicated this to be the state-of-the-art. Bidirectional Encoder Representations from Transformers (BERT), is a pretrained LM based on fine tuning. BERT LM is based on Google's attention based transformer model [1]. I followed this guide on fine tuning bert for binary text classification. I modified the method to allow for three labels, instead of two. The results of the model after training for 40 epochs on <answer-followup> pairs are:-

```
zero-one loss = 0.4924
accuracy = 0.5076
accuracy3 = 0.9091
loss = 2.4395
```

Results on <question-followup> pairs are:-

```
zero-one loss = 0.4470
accuracy = 0.5530
```

```
            accuracy3 = 0.9545
            loss = 2.4723
```

The zero-one loss of the model in both cases is quite high. This is primarily because the size of the dataset is small. With just 550 or so input examples, the model fails to learn a good distribution. And even amongst those examples, there are repeated pairs of question-followup/answer-followup that have different ratings. Even so, as the accuracy3 value shows, the model rarely classifies a question rated as 3 as 1, or vice versa. If this model were to be trained on a larger dataset, it would be a viable model to eliminate completely irrelevant questions.
The code for training the model, along with the dataset can be found here.

4 Conclusion

In trying to solve the Interview Question Generation problem, I was able to generate and evaluate questions on some level. However, there were many lessons learned along the way, which are summarised below.

1. For **Keyword Extraction**, out of the methods I tried, the RAKE algorithm generated the best results. A potential approach to this task that may yield better results is the TextRank algorithm. Moreover, the algorithm described by Tixier et al. [30] which is a method of keyword extraction using a graph-of-words approach, may produce better results as well.

2. For **Question Generation**, the best model I was able to generate was the OpenNMT model trained with ConceptNet word embeddings, and on a dataset of extracted keywords from answers. The fallacy of this model is the lack of curated training data. Among other methods tried was also a zeroshot approach to generating questions by training on Knowledge Base triples [31]. I tried to reproduce the results in the paper, but the model takes Freebase triples as input for training using the freebase api, which has been deprecated by google. Hence training and evaluating the model became impossible.

3. For the **Question Evaluation** task, which was essentially a text classification problem, I generated a model by fine tuning BERT. The inaccuracies in the predictions made by the model generated through this approach exist, once again, due to the lack of a large dataset to

train the model on. Other approaches to this task include using an n-gram model, which may use algorithms like Linear SVM, Naive Bayes, or Linear Regression. The reasons these models were not selected were:-

(a) Fine tuning BERT has been shown to achieve better results in text classification than any bag-of-words approach.

(b) The classification needed to be done with respect to relevance to a certain sentence (like the preceding answer). This is simply not possible with an n-gram model.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *CoRR*, volume abs/1706.03762, 2017.

[2] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. Incorporating copying mechanism in sequence-to-sequence learning. *CoRR*, abs/1603.06393, 2016. `http://arxiv.org/abs/1603.06393`.

[3] `http://colah.github.io/posts/2015-08-Understanding-LSTMs`.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. `http://dx.doi.org/10.1162/neco.1997.9.8.1735`.

[5] Chris Olah and Shan Carter. Attention and augmented recurrent neural networks. *Distill*, 2016.

[6] Ondrej Dusek, Jekaterina Novikova, and Verena Rieser. Findings of the E2E NLG challenge. *CoRR*, abs/1810.01170, 2018.

[7] Shuang Chen. A general model for neural text generation from structured data.

[8] Ziang Xie. Neural text generation: A practical guide. *CoRR*, abs/1711.09534, 2017. `http://arxiv.org/abs/1711.09534`.

[9] Diego Marcheggiani and Laura Perez-Beltrachini. Deep graph convolutional encoders for structured data to text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*,

pages 1–9, Tilburg University, The Netherlands, November 2018. Association for Computational Linguistics. https://www.aclweb.org/anthology/W18-6501.

[10] Sam Wiseman, Stuart Shieber, and Alexander Rush. Learning neural templates for text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3174–3187, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[11] Xinya Du, Junru Shao, and Claire Cardie. Learning to ask: Neural question generation for reading comprehension. *CoRR*, abs/1705.00106, 2017. http://arxiv.org/abs/1705.00106.

[12] Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. Neural question generation from text: A preliminary study. *CoRR*, abs/1704.01792, 2017. http://arxiv.org/abs/1704.01792.

[13] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[14] Yanghoon Kim, Hwanhee Lee, Joongbo Shin, and Kyomin Jung. Improving neural question generation using answer separation. *CoRR*, abs/1809.02393, 2018. http://arxiv.org/abs/1809.02393.

[15] Zichao Wang, Andrew Lan, Weili Nie, Andrew Waters, Phillip Grimaldi, and Richard Baraniuk. Qg-net: a data-driven question generation model for educational content. In *Proceedings of the Fifth Annual ACM Conference on Learning at Scale*, pages 1–10, 06 2018.

[16] Hady Elsahar, Christophe Gravier, and Frederique Laforest. Zero-Shot Question Generation from Knowledge Graphs for Unseen Predicates and Entity Types. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 218–228, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. https://www.aclweb.org/anthology/N18-1020.

[17] https://github.com/fastai/fastai.

[18] https://github.com/IBM/pytorch-seq2seq.

[19] https://ibm.github.io/pytorch-seq2seq/public/models.html.

[20] https://github.com/eladhoffer/seq2seq.pytorch.

[21] https://allenai.github.io/allennlp-docs/api/allennlp.models.encoder_decoders.html.

[22] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. Opennmt: Open-source toolkit for neural machine translation. In *Proc. ACL*, 2017.

[23] http://opennmt.net/OpenNMT-py/options/train.html#Model-%20Encoder-Decoder.

[24] https://github.com/moonlightlane/QG-Net.

[25] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016.

[26] Xingdi Yuan, Tong Wang, Çaglar Gülçehre, Alessandro Sordoni, Philip Bachman, Sandeep Subramanian, Saizheng Zhang, and Adam Trischler. Machine comprehension by text-to-text neural question generation. *CoRR*, abs/1705.02012, 2017.

[27] Tom Hosking and Sebastian Riedel. Evaluating rewards for question generation models, 2019.

[28] Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. *CoRR*, abs/1612.03975, 2016.

[29] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[30] Antoine Tixier, Fragkiskos Malliaros, and Michalis Vazirgiannis. A graph degeneracy-based approach to keyword extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1860–1870, Austin, Texas, November 2016. Association for Computational Linguistics.

[31] Hady Elsahar, Christophe Gravier, and Frederique Laforest. Zero-shot question generation from knowledge graphs for unseen predicates and

entity types. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 218–228, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.