QG-Net Project Report

Kancharla Aditya Hari
aditya.hari@research.iiit.ac.in

Manoj Sirvi
manoj.sirvi@research.iiit.ac.in

December 11, 2021

1 Introduction

In this project, we recreate the QG-Net model described by Wang et. al. The model has two components - a context reader that reads the given context and generates an encoding for it, and a question generator that takes these encodings as input and generates the question. Additionally, it incorporates a pointer network in the question generator to improve performance. We also experiment with a BERT based architecture for exploring possible improvements to this architecture.

2 Problem Description

The paper tackles the problem of automatic question generation for educational content. According to research, learning gives better results with frequent and quiz questions compared to spending the same amount of time reading textbooks. However because of the the rapid increase in educational content, generating questions for the content is backbreaking work if done manually. This motivates the development of a system capable of automatically generating questions from a given text which is both fluent and relevant. They propose QG-Net, a network capable of generating high quality questions after training it on publicly available general-domain question-answering datasets without any further finetuning and outperforms SoTA neural-network and rule-based systems.

QG-Net generates question by sampling question words q_t from the following probability distribution -

$$P(Q|C, A, \theta) = \Pi_{t=1}^{L^Q} P(q_t|C, A, {q_\tau}_{\tau=1}^{t-1}, \theta)$$

Here, $Q = {q_t}_{t=1}^{L}$, where L is the length of the question. Similarly C and A are the set of words in the context and answer respectively, and θ is the set of parameters

3 Dataset

Split	Size
Train	86832
Val.	5263
Test	5263

Table 1: SQuAD splits

We will be using the SQuAD (Stanford Question Answering Dataset) dataset. This dataset is a reading comprehension dataset consisting of a set of paragraphs from Wikipedia articles, an answer, and a human generated question based on the paragraph and answer. The answer to every question is either a contiguous segment of text from the corresponding reading passage. In total, this dataset contains more than 100,000 questions. For every paragraph, there can several answers and questions associated with it

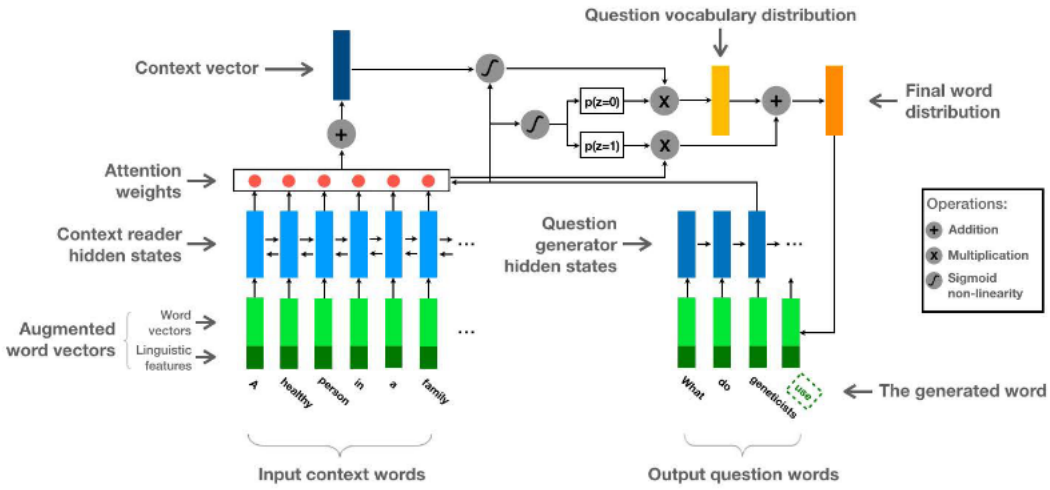

Figure 1: QG-Net Architecture (from Wang et. al)

The authors truncate each paragraph to just the single sentence containing the answer, and this is treated as the context for the model. The training, validation, and test splits are shown in table 1.

4 Implementation

4.1 Data Preprocessing

For data pre-processing, scripts provided by authors of the original paper have been used. These are publicly available on the QG-Net Github repository[1]. The data is split into source and target, and word level features related to the source are generated. These include named entity tag, PoS tag, word case and a variable indicating if a word is part of the context from which we have to generate questions.
TorchText is used for additional processing, such as generating the vocabulary and GLoVE embeddings, turning the data into tensors that can be used as input to the model, and batching the data. A batch size of 16 is used for training.

4.2 QG-Net Architecture

In our implemention, the context reader is referred to as the Encoder, and the question generator is referred to as the Decoder. The two components combined represents the complete QG-Net architecture.

4.2.1 Encoder

The encoder takes the context along with the engineered features as input and creates a representation for each word in the context. The inputs are first passed through an embedding layer to generate their embedded representation. The engineered features are - ANS which encodes whether the word is part of the answer, the part of speech tag POS, named entity tag NER, and word case CAS.

$$\tilde{c_j} = [c_j, ANS_j^T, POS_j^T, NER_j^T, CAS_j^T]$$

This $\tilde{c_j}$ is then fed to a 2-layered, bi-directional LSTM. The final hidden state of the last layers of the forward and backward LSTM is used as the representation for the j^{th} word in the context.

$$\overrightarrow{h}_j = \overrightarrow{BiLSTM}(\tilde{c}_j, \overrightarrow{h}_{j-1})$$

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

$$\overleftarrow{h}_j = \overleftarrow{BiLSTM}(\tilde{c}_j, \overleftarrow{h}_{j+1})$$
$$h_j = [\overrightarrow{h}_j, \overleftarrow{h}_j]^T$$

4.2.2 Decoder

The decoder uses a 2-layered, unidirectional LSTM network to encode the words in the question.

$$s_t = LSTM(q_t, s_{t-1})$$

During training, teacher forcing is used i.e words from the target sequence are fed to the network at each timestep, whereas during evaluation and inference, the output of the previous timestep is as input to the next.

Attention - To calculate the calculate the probability distribution over the question vocabulary, an attention mechanism is used. We calculate the attention score as follows -

$$e_t^i = v^T(W_h h_i + W_s s_t + b_{attn})$$

$$a_t = softmax(e_t)$$

A context vector is then generated by using these attention weights. Here H is the weighted sum of the hidden state representations of the context.

$$k_t = Ha_t$$

The context vector is concatenated with the question encoding and the probability distribution is computed using this. This is slightly different from the original paper, as we found that introducing non-linearity here helped in getting better performance.

$$P_1(q_t) = softmax(ReLU(W_v[s_t, k_t] + b_v))$$

Pointer - A pointer mechanism is implemented to allow generation of words from the source directly. For this, the probability of generating a word is computed.

$$p_{gen} = sigmoid(W_z s_t + b_z)$$

And the probability distribution drawn from is calculated as such -

$$P(q_t) = p_{gen} P_1(q_t) + (1 - p_{gen}) \sum_{i:w_i=q_t} a_t^i$$

4.3 BERT QG Architecture

We attempted to incorporate BERT based embeddings into the model. For this, both the encoder and the decoder were replaced with BERT models. Instead of RNN based encodings, encodings generated from BERT models are used for both reading the context and the question. The probability distribution of the vocabulary is calculated using BertLMHead provided by Huggingface[2]. The equations for the attention and the pointer probability remain the same.

However, the features used in QG-Net could not be used for this. BERT uses a WordPiece tokenizer which splits words into subwords, because of which the word levels features do not align with the BERT inputs.

5 Evaluation

To evaluate our model, we used three different evaluation metrics - BLEU, METEOR, ROUGE-L, as used in the paper. BLEU and METEOR are used for machine translation tasks, while ROUGE-L is used for summarization tasks. However, since no such metric exists specifically for question-generation, these have been used here. The results are shown in table 2.

[2] https://huggingface.co/docs/transformers/model_doc/bert#transformers.BertLMHeadModel

Model	BLEU	METEOR	ROUGE
QG-Net (Original)	0.1386	0.1838	0.4437
Our implementation	0.17	0.15	0.34
BERT QG-Net	0.08	0.12	0.22

Table 2: Evaluation scores

5.1 Examples

Some generated examples are included below.

- **Context -** A series of three small civil wars known as the Huguenot Rebellions broke out , mainly in southwestern France , between 1621 and 1629
  Question - what was the name of the french civil war ?
- **Context -** A teacher 's role may vary among cultures . teachers may provide instruction in literacy, craftsmanship or vocational training , the arts , religion , civics , community roles , or life skills
  Question - what is one of the subjects that can be used to teach education ?
- **Context -** A site was chosen in Houston , Texas , on land donated by Rice University , and administrator Webb announced the conversion on September 19 , 1961
  Question - when was the houston building completed ?

5.2 Analysis

The QG-Net implementation obtains good results. The scores are close to the original paper, and the slight discrepancy can be explained due to the fact that our model was trained for only 10 epochs due to computational limits, compared to the 20 epochs of the original paper. The BERT model was expected to perform better than the QG-Net. However, due to computational constraints this could not be realized. The model requires significant time and memory to train, which was not available at our disposal. The model could be trained for only a few epochs on a truncated version of the dataset, because of which the performance is significantly worse.