END-TO-END QUESTION GENERATION MODEL USING DEEP LEARNING

Dheeraj Jeevagasami
dj78@iu.edu

Bhawna Mishra
bhmish@iu.edu

Sumitha Vellinalur Thattai
svtranga@iu.edu

ABSTRACT

Assessments are an integral part of any education system. Exams are a way to measure a person's knowledge and skill in any subject. Manually creating questions is a difficult task that involves knowledge, expertise, and resources. In general, professors invest a lot of time in framing questions and making sure it aligns well with the course syllabus. Furthermore, some competitive exams require translating text from one language to another to make sure the questions like text comprehension are unique. Our project aims to solve these problems by translating the given text into English and then creating a summary of the text. Additionally, our system will also automatically generate multiple-choice questions from the translated and summarized text. We used transformers with multi-head attention to translate the text. Furthermore, encoder-decoder model with single-head attention was used to summarize the text. Finally, we used BERT to generate keywords and word-to-vector to generate distractors for multi-choice questions.

1 OUR SYSTEM

Our system takes German articles as input. The text translation model trained using transformers with multi-head attention translates the German text into English. We then use the model trained using encoder-decoder with attention to create a summary of the English article.

The summarized text is then used to generate keywords and distractors for the multiple-choice questions. For keyword generation, we have used the pre-trained model BERT, and we have used Glove word2vec to find similar words for distractors.

1.1 BLOCK DIAGRAM

Below is the block diagram describing the end-to-end process:

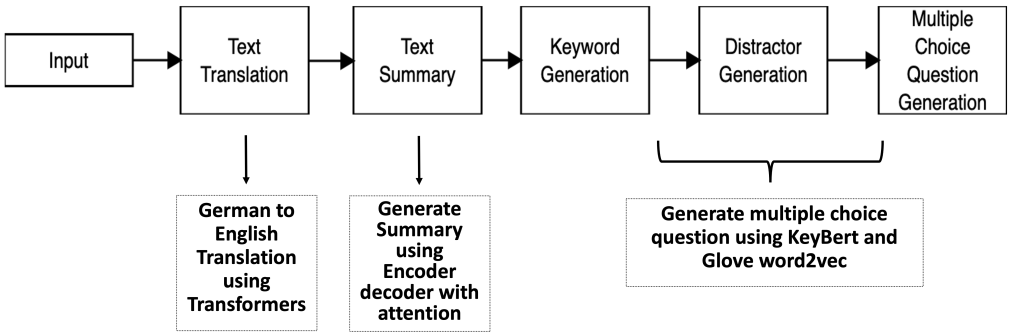

Figure 1: Overall System

2 MODEL

This section describes the model architecture for the different processes in our system:

2.1 TEXT TRANSLATION

We used the Multi30K dataset for training and testing the model. The dataset has about 29K training samples and 10K testing samples. The average length of an English sentence is 13 and in German is 12.4.

The dataset can be accessed from - here

The metric used for evaluation is BLEU score (BiLingual Evaluation Understudy). BLEU score is obtained by multiplying the Brevity Penalty with the Geometric Average of the Precision Scores.

$$BLEU = Brevity * GeometricAverageofthePrecisionScores \tag{1}$$

Brevity - It is a penalty score that penalizes the small sentences

Precision - It is the number of words in the predicted sentence that also occur in the actual sentence

2.1.1 ENCODER-DECODER ARCHITECTURE

The sequence-to-sequence model also known as the encoder-decoder model trains 2 recurrent neural networks built using GRU. The encoder, the first RNN, is taught to take German text and encode it in sequential order. After receiving the encoded sequence, the second RNN decoder conducts a mapping to the text and produces the English text.

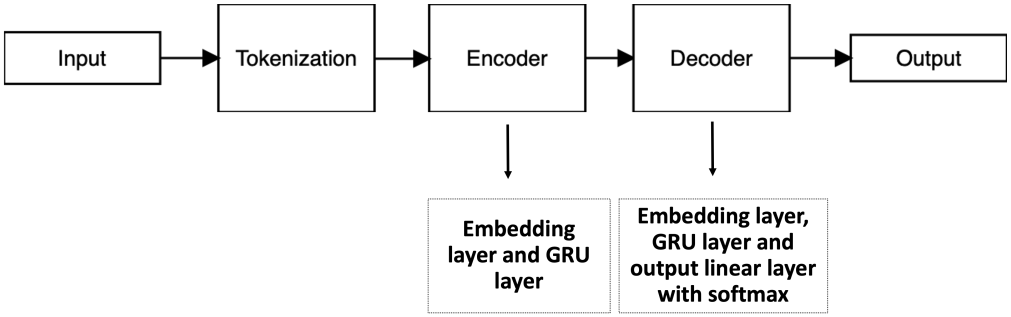

Figure 2: Block Diagram of Encoder-Decoder Model

The input is the german text which is taken from the Multi30K dataset. The text is tokenized and fed as input to the encoder. The encoder is built using Gated Recurrent Unit (GRU) which is more efficient than LSTM in handling memory and has very similar performance.

The output of the encoder is then fed into a decoder which is also built using GRU. Adam was used as an optimizer with a default learning rate of 0.001. Gradient clipping which is a technique to prevent exploding gradients in RNN was also used to improve the performance.

Insights:

We trained the model with 256 and 1200 hidden units for 35 epochs.

- The BLEU score using 256 hidden units was 0.079 and using 1200 was 0.118. The performance of the model did not improve further with the increase in hidden units
- Since the performance is not expected, we improved the performance of the model using transformers with multi-head attention model

2.1.2 TRANSFORMER WITH MULTI-HEAD ATTENTION

To determine the significance of each word in the sentence, the transformer model uses a self-attention process to extract the features of each word.

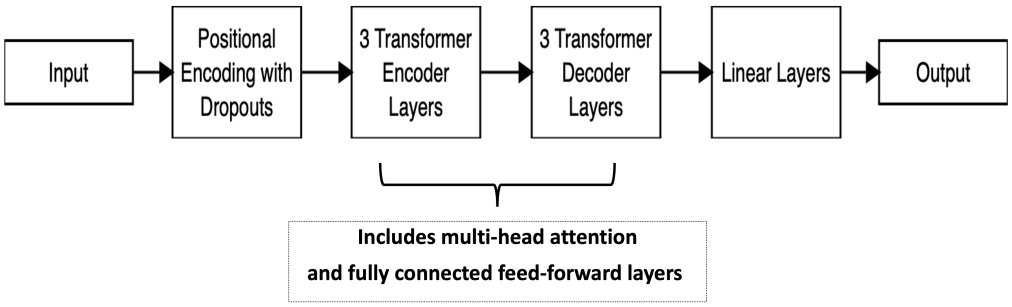

Figure 3: Block Diagram of Transformer Model

The german text is tokenized and a vocabulary is created. This tokenized text is given as input to the encoder. Token embeddings are used to express the transformed text into tokens. To obtain the notions of word order, the positional encoding function is introduced to the token embedding.

The above model also consists of an encoder and a decoder. However, the way they are built is different. In this model, the encoder and decoder contain a core block of attention and a feed-forward network repeated 3 times.

The encoder is made of a multi-head self-attention layer and a fully connected feed-forward network. The decoder contains three sub-layers, two multi-head self-attention layers, (one of the multi-head self-attention layers is used to perform self-attention over encoder outputs), and a fully connected feed-forward network. The output is then fed to a linear layer which gives the translated English text. Layer normalization is included in both the encoder and decoder.

The above model was trained with 256 hidden units for 35 epochs. Adam was used as an optimizer with a learning rate of 0.0001.

Below is the performance comparison of the different models:

Model	BLEU Score
Encoder - Decoder (Hidden units - 256)	0.079
Encoder - Decoder (Hidden units - 1200)	0.118
Transformer (Hidden units - 256)	0.246

Figure 4: Performance Comparison of Different Models

Insights:

- The model using transformer architecture trains faster and converges to a lower loss compared to the previous model
- The model performed significantly better than the normal encoder-decoder model with a BLEU score of 0.244

2.1.3 RESULTS

Below is the training and validation loss for the transformer model:

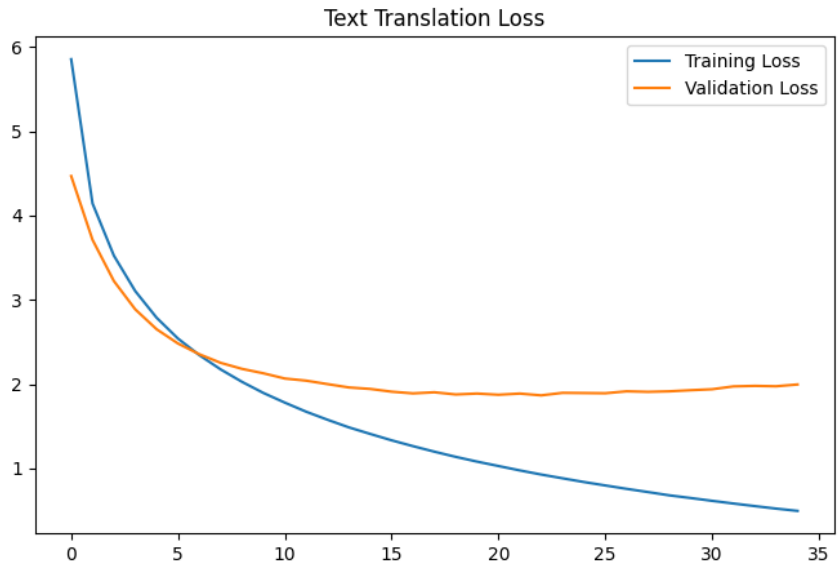

Figure 5: Train and Validation Loss of Transformer Model

Below shows the actual and predicted text using the best model:

Actual	Predicted
a sleek yellow dog mid run on a dirt area on an obstacle course	a lean dog is tan dog pushing a rock covered sculpture while set up in a area of dirt . \n
a black and white photo of a dog staring ahead with a background of signs written in asian language characters .	a grand guard is looking out at the water with a signs in an asian sign . \n
an aerial landscape photo of browning grasses and a castle-like structure peaking through and a mountain landscape in the horizon .	an amazing rock climber covered in brown , an open building , which is inside the horizon and a mountain landscape near the horizon . \n
a bare-chested man belts out a song during a stage performance .	a naked man with bare feet performing a song on a stage . \n
a slim asian man in a long white apron and blue hat stringing pasta in a restaurant kitchen .	a lean , asian man in a long white apron and blue apron pulls a restaurant inside a restaurant . \n
a man and a woman walking in the city	a man and woman walking in the city . \n
a waterfall in a forest with many trees	a waterfall in a forest with many trees . \n
a bee hovering over purple and orange flowers .	a bee climbing over snow covered flowers . \n
a calm lake surrounded by trees and rocks .	a rock climber , lifting some rocks and rocks on the lake . \n
two people on a football field one on the ground .	two people on a soccer field , one on the ground . \n

Figure 6: Translation Result Using Transformer Model

From the table, we see that the translation is not accurate and hence we will use the pre-trained BERT translation model for our prototype question generation.

2.2 TEXT SUMMARIZATION

The amount of textual data being produced every day is increasing rapidly both in terms of complexity as well as volume. In most cases, we might want to summarize the content to identify the most important parts of the content. With the advancements in Deep Learning, we can build models to shorten long pieces of text and produce a crisp and coherent summary to save time and understand the key points effectively. Text summarization can be broadly classified into two parts:

2.2.1 EXTRACTIVE SUMMARIZATION

In this type of summarization, we formulate a summary by identifying the most important and crucial parts of the text. This method of summarization can be achieved through text ranking. The process involves weighing the words/tokens in the text and giving the importance scores. Extractive summarization displays the most important parts of the content and will not rephrase the actual content. Hence, we will get an excerpt from the text. We have used BERT models to display important parts of the passage as a summary.

Below is a simple representation of how extractive summarization works:

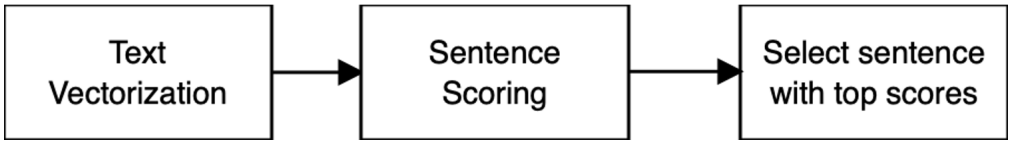

Figure 7: Extractive Summary

2.2.2 ABSTRACTIVE SUMMARIZATION

In this type of summarization, we take the entire text as input and generate an overall summary of the content typically using deep learning techniques. The advantage of abstractive summarization over extractive summarization is that it considers all parts of the content and hence there is a comparatively minimal data loss as compared to extractive summarization. We have used an encoder-decoder deep learning model along with single-head attention to implement abstractive summarization. This technique involves the generation of entirely new phrases that capture the meaning of the input passage.

Below is a simple representation of abstractive text summarization.

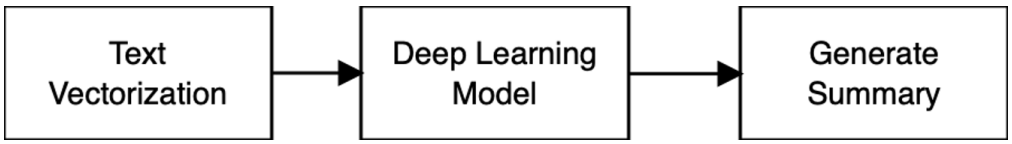

Figure 8: Abstractive Summary

We have used the CNN Daily news articles dataset for training our model. Our dataset consists of over 200,000 daily news articles along with their summary.

The encoder is built using 3 LSTM layers. The output of the encoder is then fed into the decoder which is also built using LSTM. The decoder embeds the input and passes it on to the attention layer. Attention is applied by using batch matrix multiplication of the attention weights and the encoder output. The output of this attention layer is passed on to the LSTM layer. The output of the model is a summarized text.

The model was trained for 50 epochs. Adam was used as an optimizer and sparse-categorical entropy was used as the loss function.

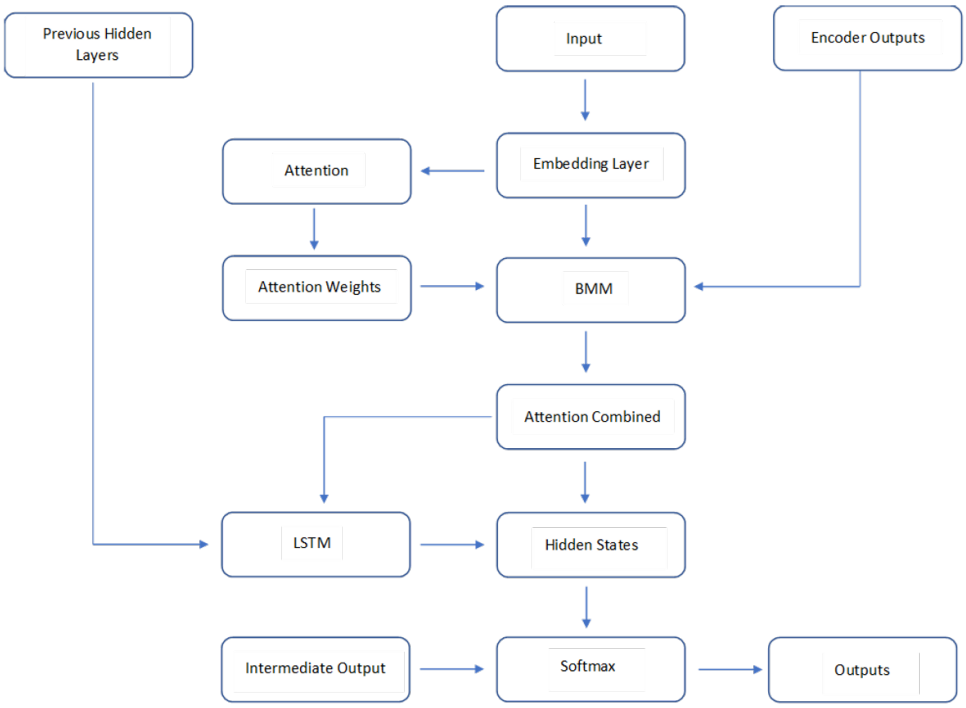

Figure 9: Block Diagram of Text Summarization Model

2.2.3 RESULTS

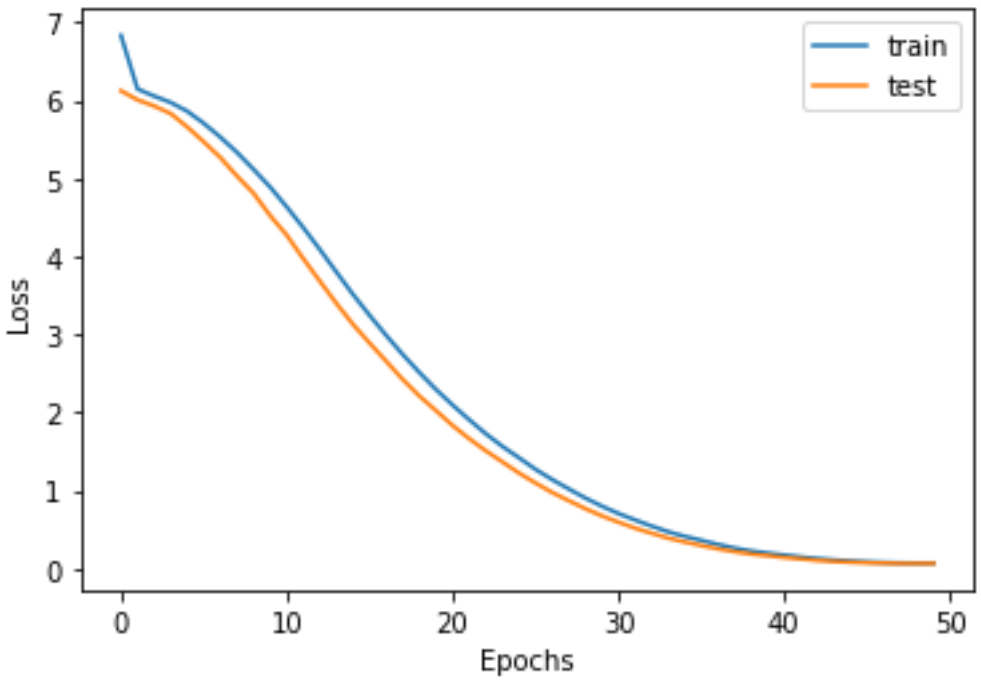

Figure 10: Loss plot for train and test sets using categorical cross-entropy loss

id	original_summary	predicted_summary
0	bishop john folda north dakota taking time diagnosed contracted infection contaminated food italy church members fargo grand forks jamestown could exposed	cafe minister fargo another contaminated morning family data sults port isle incident squad teenager goal forks hearing jamestown assassins people keeping says sostok sostok sostok sostok sostok ...
1	criminal complaint used role help cocaine traffickers ralph mata internal affairs lieutenant allegedly helped group guns also arranged assassins murder plot complaint alleges	crown larger south evidence state tested nina lieutenant trick raising arranged hispanic came risk arrested plot years alleges eccleston head wore larger bradford says sostok sostok sostok sostok...
2	eccleston todd drunk least three pints driving using phone veered across road yarmouth isle wight crashed head year rachel titley died hospital police would legal drink drive limit time crash foun...	news home called forces drive road drive lived thought residents road drive road poyet found player trial lived drive found family factories says sostok sostok sostok sostok sostok sostok sostok ...
3	nina santos says europe must ready accept sanctions hurt sides targeting russia business community would sapping support president putin says says europe would hard time keeping factories going wi...	laboratory draws year business calls target bigger sarah gaal firms enjoying role prime talks time scunthorpe friends told performance year year business time beach family counter fleetwood owner...
4	league scunthorpe peterborough bristol city chesterfield crawley drop first points season stand striker matt done scores trick rochdale thrash crewe wins notts county yeovil coventry bradford oldh...	suggests released perception telford passenger much yeovil ship jack passenger agreement comment confirmed lawyers days position front days suggests firm vale firm debut issues tough says sostok ...

Figure 11: Output of the Summarized Text

As we can see the results of the abstractive summary model are not very good even though the training and validation loss is decreasing. Hence for our prototype model, we will be using our results from the extractive summarization to generate questions.

2.3 Question Generation

The goal of automatic question generation is to generate questions from a context, with answers corresponding to the passages given. We started by using the TextRank algorithm which uses cosine similarity to identify important sentences and create an extractive summary. However, the BERT, which builds on the self-attention transformer network architectures worked better. Additionally, we used the KeyBERT keyword extraction technique which is an easy-to-use Python package for extracting keyphrases using BERT embeddings to find keywords and key phrases that are most similar to a passage. Using this as our input, we generated a more semantically consistent and communicative question.

Overall, for the prototype model, we used the pre-trained BERT model for translation and summarization. This summary will then be used to generate questions.

2.3.1 Block Diagram

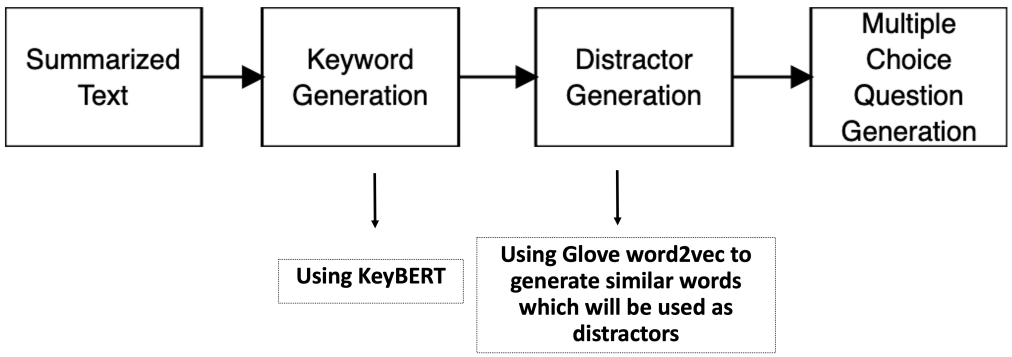

Figure 12: Block Diagram of Question Generation Model

2.3.2 KEYWORD GENERATION

The first step of Question Generation is Keyword Generation using the KeyBERT model. We have provided the summarized text as input and have used the sentence transformers (”all-mpnet-base-v2”) as it works great for English phrases and gives the best quality. We have set the ’keyphrase-ngram-range’ to (1, 1) to set the number of words in the resulting keyphrases. Using this, we were able to see that the model was indeed able to identify words that appeared to reflect the meaning of the given text.

2.3.3 DISTRACTOR GENERATION

In the next step of the Question Generation, as input, we will take the summarized text and a set of questions and their correct answers to generate several false answers that relate somehow to the answer and are consistent with the semantic context of the question. For creating the distractors, we have used the Glove word2vec algorithm which is an unsupervised learning algorithm used for obtaining vector representations for words. Using GloVe we make sure that the vector differences capture as many meanings specified by the juxtaposition of two words as possible. After generating the distractors, we generate the multiple-choice questions by providing a mix of keywords and distractors as the possible options for answers to the multiple-choice questions.

2.3.4 RESULTS

The left-hand side shows the keywords generated for a sports article along with the importance. The right-hand side is the question generated for the same paragraph. The options were generated using distractors (Glove word2vec)

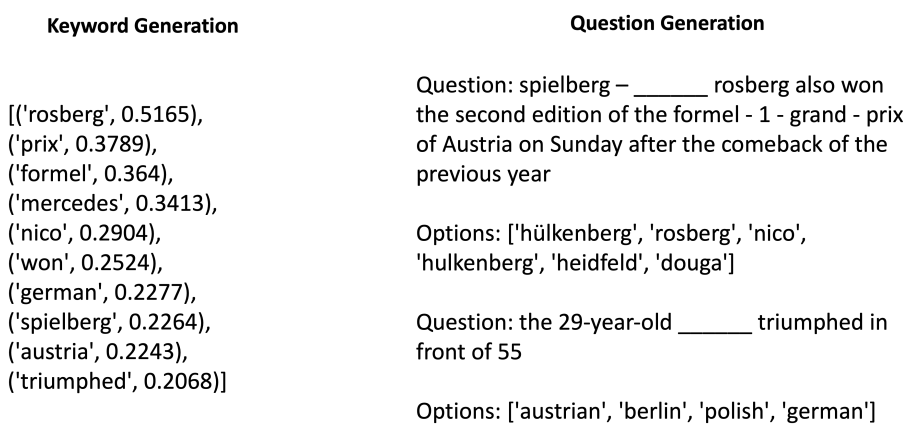

Figure 13: Results of Question Generation

3 LIMITATIONS

- BLEU score of the translation model using a normal encoder-decoder model was very poor and was improved by using transformers. However, there is still room for improvement. The performance can be improved by using pre-trained models
- Summarization model does not generate accurate summaries. Its performance can be improved by using transformers like GPT2 and T5
- Currently, the question generation uses only the summarized text to generate questions. The process can be improved by using hugging face models and BERT which will generate questions that are more accurate and more challenging.

4 Conclusion and Future Works

4.1 Conclusion

We tried to generate questions to eliminate the manual overhead of creating questions for comprehensive passages. We built a translation model using the encoder-decoder model and also compared its performance with transformers. The transformer with multi-head attention indeed performed significantly better than the encoder-decoder model. Additionally, for abstractive text summarization, we used the encoder-decoder model with attention. Since the performance of the two models was not good to feed as an input to the question generator, we used BERT to perform the task for this prototype.

We used BERT with sentence transformers on our input - German sentences, to perform text translation and convert the German text into English and then used this translated text to summarize the English text. This summarized text was then provided as input to generate keywords using BERT which will be used as a blank in our multiple-choice question. Then we generate distractors using the GloVe word2vec algorithm to finally create the multiple-choice questions with a mix of keywords and distractors as possible answer choices. The model we used can generate questions that are generally understandable to humans, but with certain limitations with scope for more improvements.

To develop questions that are streamlined to the particular article and make sense every time, our models (translation and summarization) require some additional fine-tuning. We plan to continue working on this project to improve the quality of the questions and get more knowledge on how to develop an end-to-end machine-learning pipeline.

4.2 Future Works

- Improve the performance of all tasks by using any pre-trained models and building on top of it.
- We will use the abstractive summary as an input to question generation rather than extractive summary since we will be able to generate more challenging questions using an abstractive summary
- Extend question generation to different question types – True/False, Open-ended question

Acknowledgments

We thank Professor Henry Choi and the TA's for being a great support during the entire semester. It was a great learning experience and we look forward to tuning this project to improve its performance.

References

[1] Ying-Hong Chan and Yao-Chung Fan. 2019. A Recurrent BERT-based Model for Question Generation. In Proceedings of the 2nd Workshop on Machine Reading for Question Answering, pages 154–162, Hong Kong, China. Association for Computational Linguistics.

[2] De Kuthy K, Kannan M, Santhi Ponnusamy H, Meurers D. Exploring neural question generation for formal pragmatics: Data set and model evaluation. Front Artif Intell. 2022 Oct 31;5:966013. doi: 10.3389/frai.2022.966013. PMID: 36388400; PMCID: PMC9661521.

[3] A. R. Fabbri et al. "Template-Based Question Generation from Retrieved Sentences for Improved Unsupervised Question Answering". In: ACL. 2020.

[4] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692, 2019.

[5] Klein, T., Nabi, M. (2019). Learning to Answer by Learning to Ask: Getting the Best of GPT-2 and BERT Worlds. ArXiv, abs/1911.02365.

[6] Vishwajeet Kumar, Ganesh Ramakrishnan, and Yuan-Fang Li. 2018. A framework for automatic question generation from text using deep reinforcement learning. arXiv preprint arXiv:1808.04961.

[7] Dmytro, Kalpakchi, Johan, Boye (2021). BERT-based distractor generation for Swedish reading comprehension questions using a small-scale dataset. arXiv:2108.03973

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

[9] Stéphane Clinchant, Kweon Woo Jung, Vassilina Nikoulina. (2019) On the use of BERT for Neural Machine Translation. arXiv:1909.12744

A Appendix

A.1 Other online Reference Resources

- PyTorch Translation Tutorial
- PyTorch Language Translation using Transformer
- Text Summarization Using TextRank, Seq2Seq, and BERT
- Question generation using NLP
- Text Summarization Using Deep Learning

A.2 Referenced GitHub Repositories

- Sequence to Sequence Model with Attention

A.3 GitHub

- Link to the code: GitHub Repo

A.4 YouTube Video Link

- Link to the Video: YouTube Video