Multiple-choice content generation for practical assessment of students' knowledge

Project Report

Haonan Liu

Student ID: 882794

Vinh Nguyen

Student ID: 732336

Germans Savcisens

Student ID: 970059

A report is prepared as a part of the
Statistical Natural Language Processing course

Aalto University
29th April 2021

1 Introduction[1]

Standardised tests have been long an essential part of the admission process to various educational institutions. They measure essential and nontrivial skills and determine whether they align well with the requirements. To that end, the development of test comes with the price of expensive intellectual labours and time-consuming tasks, not to mention possible bias.

To that end, various committees, including academic advisors and test developers, are actively involved throughout the test development procedure to ensure the highest-quality assessments. It can come with the price of expensive intellectual labours and time-consuming tasks, not to mention possible bias. Thus, there is a need for an improvement for the test development pipeline.

We intend to simplify the process by using recent research within natural language processing, namely techniques for sequence processing with deep neural networks. Our goal is to build an *end-to-end system* that can *generate content for multiple-choice questions*. Thus, given the paragraph and desired *correct* answer, the system should provide the corresponding question, as well as the set of possible *incorrect* answers[2].

> **Our code is located here**: `https://github.com/vinhng10/decepticon/`

2 Methods

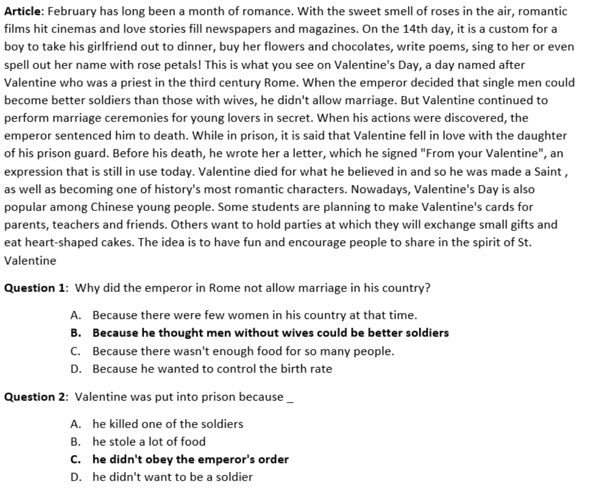
The following section describes the model and methods we use during the project. We start the section by presenting the dataset; then, we describe the theoretical background for the question generator and the distractor generator. We end the chapter with a description of evaluation techniques for QA pairs.

Our project aims to utilise several common techniques such as `RNNs`, `BERT` and `T5` to find the most optimal solution.

2.1 Data

Our focus domain is examination (or assessment) of students' knowledge. Thus, we use ReAding Comprehension Dataset From Examination or `RACE` (Lai et al. 2017) for our project. The dataset consists of middle and high school exam multiple-choice questions. It contains around 28000 unique articles and 100000 unique question-answer pairs. The example of a data sample is displayed in Figure 1.

The dataset is already split into train, validation and test sets. Thus, we can compare the resulting metric to other existing solutions. Our preliminary data analysis on the training set[3] shows that articles on average have 280 tokens (with a maximum of 1162). When it comes to the questions, the average number of tokens per question is 10 (with a maximum of 63). For correct and incorrect answers, the average number of tokens is 5 (maximum: 105). Per one article in the train set, we have, on average three questions (maximum: 7). The dataset contains several types of questions: *wh-questions*, *cloze-style* questions, and **confirmation of true/false statements**. It should be noted that the majority of questions are on the form *In this article, we have learned that _*, thus, *cloze*-questions are over-represented.



Article: February has long been a month of romance. With the sweet smell of roses in the air, romantic films hit cinemas and love stories fill newspapers and magazines. On the 14th day, it is a custom for a boy to take his girlfriend out to dinner, buy her flowers and chocolates, write poems, sing to her or even spell out her name with rose petals! This is what you see on Valentine's Day, a day named after Valentine who was a priest in the third century Rome. When the emperor decided that single men could become better soldiers than those with wives, he didn't allow marriage. But Valentine continued to perform marriage ceremonies for young lovers in secret. When his actions were discovered, the emperor sentenced him to death. While in prison, it is said that Valentine fell in love with the daughter of his prison guard. Before his death, he wrote her a letter, which he signed "From your Valentine", an expression that is still in use today. Valentine died for what he believed in and so he was made a Saint , as well as becoming one of history's most romantic characters. Nowadays, Valentine's Day is also popular among Chinese young people. Some students are planning to make Valentine's cards for parents, teachers and friends. Others want to hold parties at which they will exchange small gifts and eat heart-shaped cakes. The idea is to have fun and encourage people to share in the spirit of St. Valentine

Question 1: Why did the emperor in Rome not allow marriage in his country?

A. Because there were few women in his country at that time.
B. Because he thought men without wives could be better soldiers
C. Because there wasn't enough food for so many people.
D. Because he wanted to control the birth rate

Question 2: Valentine was put into prison because _

A. he killed one of the soldiers
B. he stole a lot of food
C. he didn't obey the emperor's order
D. he didn't want to be a soldier

Figure 1: Example of the article-question-answers pair from the `RACE` dataset. Each article has multiple associated questions. Each question has four associated answers. The **highlighted** answers are correct, the rest are not.

Question-answer pair cover a diverse set of tasks such as (Lai et al. 2017):

- **Reasoning**: the answer can be deducted from the single/multiple sentences in the articles,
- **Paraphrasing**: the answer is a rephrased version of an existing sentence,
- **Matching**: the question is the exact copy of a sentence with the missing words or word spans,
- **Obscure tasks**: the answer is not given in the article.

From the list above, we see that dataset contain the QA pairs of different complexity and task. Hence, it is a well-grounded benchmark for the goal we want to achieve.

2.2 Question Generation

For question generation we consider three models: `Seq2Seq`, `BERT` and `T5`.

2.2.1 Seq2Seq model

The first model is a typical Sequence-to-sequence (`Seq2Seq`) model. Multiple works have already adapted recurrent neural networks for question generation (see Zhao et al. (2018), Liu et al. (2020); thus, it is a perfect choice for the baseline models. `RNNs` for text generation only use two recurrent neural networks: encoder and decoder. The encoder processes the input sentences to establish a context vector (i.e. latent representation of the input), while the decoder converts the vector into output sentences. Since *vanilla* `RNNs` have issues with exploding and vanishing gradients, we focus on applying gated units, namely the **Gated Recurrent Unit**, `GRU` (Goodfellow et al. 2016). `GRU` has an advantage over the other gated architecture, i.e. `LSTM`, as we need fewer parameters to tune (Goodfellow et al. 2016) to achieve similar performance.

2.2.2 BERT-based model

Our second model is based on the **Transformer architecture** (Vaswani et al. 2017). The core idea behind Transformers is the **attention** mechanism that

[1] This section contains several parts from our Literature Review report, as well as, the Project Proposal report

[2] further, we call them distractors

[3] training set contains 25135 unique articles

attends to different positions in the sequence to derive the representation of the sequence. The attention mechanism of Transformers contains *scaled dot product attention* (which is then combined in *Multihead attention*). It allows Transformer-based models to tackle the problem of forgetting long-distance dependency, as well as enhancing the generation task's performance.

BERT is a version of a transformer with only the encoder part (Devlin et al. 2018). So far, we have seen multiple QA frameworks based on BERT (Chan & Fan 2019) (and we can see that these models are superior to the RNN-based models). BERT is a powerful tool to derive the representation of the input sequences (Devlin et al. 2018); it is not suitable for text generation. Hence, we need a unit that can decode the representation (similarly to RNN decoder).

Thus, the full model consists of the **backbone** (a BERT model) and the **generation head**. The latter is a Transformer's decoder layer. We hope to outperform Chan & Fan (2019) since they did not use a transformer-based decoder (instead, to generate sequences, they apply BERT recurrently)

2.2.3 T5-based model

The third model is also based on the Transformer architecture. T5 model (Raffel et al. 2019) is specifically designed to handle text-to-text tasks; thus, it is superior to the BERT, when it comes to text generation. A similar approach is used for quiz-question generation in Lelkes et al. (2021).

As BERT, T5 is pre-trained on the Masked Language model task, plus a Next Sentence Prediction task (Raffel et al. 2019). It is then possible to take the pre-trained model and fine-tune in on a downstream task (in our case, question generation).

2.3 Search

To generate output sequences (i.e. questions), we focus on three autoregressive search techniques: **Beam Search** (Jurafsky & Martin 2020), **Top-K Sampling** (Holtzman et al. 2020), and **Top-P Sampling** (Fan et al. 2018). Beam search allows us to traverse through many possible paths without looking at every path in the graph. At each step, beam search keeps the k-paths that have the highest probability. It expands most probable paths further and again keeps only k most probable ones.

Sampling is a slightly relaxed version of the beam search, where we do not constrain choosing **the most probable token**. Instead, we randomly sample next words from the distribution of tokens, i.e. we allow for low-probability tokens. As such, we might get a more diverse set of output sequences. In the case of the top-k sampling, we limit the number of possible tokens to the top-k ones (Fan et al. 2018). The top-p sampling restricts the sets of words so that the sum of their probability is larger than p.

2.4 Question Filtering

Even with finetuning and searching, we expect our model to *miss-fire* and generate questions that poorly correlate with answers or paragraphs. In order to tackle the issue, we use the method introduced by Liu et al. (2020). The idea is to use another finetuned transformer (namely, BERT) to filter out the weak questions. The method calculates the quality score for each generated question allowing us to choose the one that has the highest score. It will not necessary choose the most meaningful sequence, but it was still shown to improve the quality of question-answer-article pairs (Liu et al. 2020).

2.5 Distractor Generation

Another important aspect of multiple-choice questions is incorrect options or **distractors**. Gao et al. (2019) has implemented the LSTM-based model for distractor generation. The idea is to pass answer along with question and article to generate a set of incorrect answers. Not many works have tried to adapt transformers for this task. Thus, we might take advantage of BERT and T5 models to create the more powerful generator.

2.6 Evaluation Metrics

For quantitative assessment, we use BLEU, METEOR and ROUGE scores. These metrics are used to evaluate many state-of-the-art question generation models (Scialom et al. 2019, Lelkes et al. 2021, Liu et al. 2020).

BLEU metric (Jurafsky & Martin 2020) is based on the counting of n-grams appearances. The idea is to count the number of times n-gram appears both in the target and prediction sequences. This method accounts for cases when the length of a predicted sequence is different from the target.

On the other hand, it also accounts for cases when predicted sequence has multiple repeated n-grams (that appear in the target text). BLEU is a metric to evaluate the whole corpus, not single sequences (Jurafsky & Martin 2020).

Another n-gram based metric is METEOR (Denkowski & Lavie 2014). To calculate the score, we match and align the unigrams between target and predicted sequences. It allows us to overcome the issues of BLUE that looks at n-grams locally and fails when target phrases are separated and placed in different parts of the predicted sentences (Jurafsky & Martin 2020).

ROGUE metric utilises the embeddings of tokens (Ganesan 2018). Instead of matching n-grams, it finds embeddings with the smallest distance (again, between target and predicted sequences). As such, it does not look for identical matches, rather than closely related *concepts*. Thus, if a target sentence contains the word *reasonable* and the model generates sequence with the word *rational*, ROUGUE metric would give a higher score (as those are synonyms). In the case of BLEU and METEOR, it would ignore the match.

To sum up, we will use the mentioned metrics to have a valid and meaningful comparison and assessment of the models we develop.

As we now established the theoretical context for the project, we now move to the experimental setup.

3 Experiments

The following section describes the pipeline for the question generation and distractor generation training and evaluation.

3.1 Input and output

For question generation, we feed models with the concatenated text: the correct answer and the content of the corresponding article. To help the model separate the two, we use special tokens [ANS] and [CON]. In BERT and T5, these tokens are added to the pre-trained embedding space (during the finetuning, the model learns the representation). The form of the input can be seen below:

```
[ANS] (answer tokens) [CON]
(context tokens)
```

To generate distractors, we feed models with the concatenated text (question + context). To separate the two, we add a special token for the questions `[QUE]`. The input is in the same form as above; we only change `[ANS]` to `[QUE]`. For all models, we use **AutoTokenizer**[4] with the appropriate configuration (i.e. specific for `tiny BERT` and `T5-small`). It is used for encoding and decoding the input-output sequences.

3.2 Tuning

In order to tune the hyperparameters, we use the **Ray package** package (Liaw et al. 2018). We use **Hyper-Opt** (Bergstra et al. 2013) optimizer with the **ASHA** (Li et al. 2018) scheduler.

For the `RNN` model, the tuning is done on 25% of training data. Each setup is evaluated for a maximum of 7 epochs. The goal is to find the model that achieves the lowest loss. We look over the combination of batch size, a number of `GRU` layers, embedding size and hidden layer size, and bi-directionality. Tuning is not performed on `BERT` and `T5` models since we do not have access to a large amount of computational resources.

The configuration for the **sampling methods** (namely, a combination of values for **Top-p** or/and **Top-k**)[5] is tuned on the validation set. Here, we look for the configuration that maximizes `BLEU-1`, `BLEU-2`, `BLEU-3`, `BLEU-4`, `METEOR` and `ROGUE` (i.e. take the sum of all metric scores).

3.3 Question Generation

The following subsections describe the general setup and training procedure for the question generation models.

3.3.1 Generation with Seq2Seq

Our RNN consists of encoder-decoder GRUs with hidden size of 780 and embedding size of 578. The model is trained with AdamW optimizer with the learning rate of $1e-4$ and batch size of 32. During training, a *teacher forcing technique* is employed to guide the decoder to generate better output.

3.3.2 Generation with BERT

Due to the limitation in hardware capacity, we only use small scale pre-trained model. The backbone of the model is the **pre-trained BERT**[6] (pretrained on BookCorpus and English Wikipedia), which only contains two transformer encoder layers. The generation head of the model is a layer of a transformer decoder. For training we use `AdamW` optimizer with the learning rate $1e-4$. With small backbone, a batch size of 32 was used to speed up the training process.

3.3.3 Question generation with T5

We use pretrained `T5-small`[7] model. Before finetuning we add special tokens to the existing embedding space of the `T5` model: `[ANS]`, `[QUE]`, `[CON]`. The model then learns the representation of these tokens during the training phase.

To finetune the model, we use `AdamW` optimizer with the learning rate of $1e-4$ (and Multiplicative Learning Rate Scheduler with the learning rate decay of 0.05 per epoch). We also apply a weight decay of $1e-4$ on weights (excluding biases and parameters of Layer Normalization units). To speed up the training and lower the computational complexity we use $16-bit$ precision floats. The size of each batch is 12, we accumulate loss over 10 batches (we artificially provide a batch of 120 to compute a loss at each step).

We use an **Early Stopping** to terminate finetuning if the *Cross Entropy Loss* on validation set does not decrease for 3 epochs (with minimum decrease of 0.05 units). And we use **Gradient clipping** to avoid the explosion of the gradient (with value of 5).

We do not finetune the available hyperparameters for this model, since it is computationally heavy.

3.4 Quality Judge

To assess the quality of question-answer pairs we use the finetuned BERT model [8]. By feeding in the generated questions together with the correct answers, the model assigns *quality* scores to the pair. According to the documentation, it checks the overall correspondence of question-answer pair, i.e. it does not necessary mean that answer is a correct response to the question. The model is already finetuned on `SQuAD`, `CoQA`, `MSMARCO` and `RACE` (i.e. test set), so we do not have to perform any additional finetuning. We implement Quality Judge (`QJ`) on the top of the best performing model (out of our three).

The idea is to generate multiple question sequences per answer, and then choose the question that has the highest score. Only the chosen questions are going to be used to calculate the metrics. The model takes the standard `BERT` form of an input:

```
[CLS] question tokens [SEP]
answer tokens
```

3.5 Distractor Generation

In this experiment, we use the same architecture and configuration for `Seq2Seq`, `BERT`, `T5` models as described in section 3.3 for the distractor generation task. The only difference is the form of the input, which now looks like

```
[ANS] correct answer tokens
[CON] context tokens [QUE]
question tokens
```

For each question-answer-article pair we have three distractors in our dataset. During the training, we randomly pick one of the distractors, which becomes our target.

3.6 Evaluation

We conduct both quantitative and qualitative assessment of the performance of four models. In quantitative assessment, we compute 6 metrics mentioned above including `BLEU-1`, `BLEU-2`, `BLEU-3`, `BLEU-4`, `METEOR`, and `ROUGE-L` for each of the models on the test dataset. This assessment is fast to conduct, but there are evidences that these automatic assessment do not correlate well with the human evaluation (Callison-Burch et al. 2006). Therefore, quality

[4] https://huggingface.co

[5] RNN has only Top-p sampling

[6] https://huggingface.co/prajjwal1/bert-tiny

[7] https://huggingface.co/t5-small

[8] https://huggingface.co/iarfmoose/bert-base-cased-qa-evaluator

Table 1: Experimental results of our models on question generation

	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L
Seq2Seq	0.08	0.03	0.02	0.00	0.09	0.13
BERT	0.18	0.09	0.03	0.04	0.14	0.19
T5	0.29	0.19	0.13	0.09	0.28	0.32
T5 + QJ	0.31	0.21	0.13	0.09	0.28	0.33

assessment is needed, that is, we manually generate samples from the models and evaluated them.

In order to compare the generated samples from the models, we pick a set of inputs which included articles and correct answers and covered a various question types. This input set is then fed to each of the model to extract the token ID sequences. From there, we use the pre-trained tokenizers that are used for each model to decode those sequences into the text sequences.

4 Results

The following chapter describes results and findings that were produced during our experiments.

4.1 Evaluation of question generators

After training, we ran all three models on held-out test set and compute the metrics to compare their performance.

EQG-RACE (Jia et al. 2020) studied similar problem of question generation. However, the authors remove *cloze* and *general question*[9], and only used specific question. Therefore, we could not include this work for benchmark.

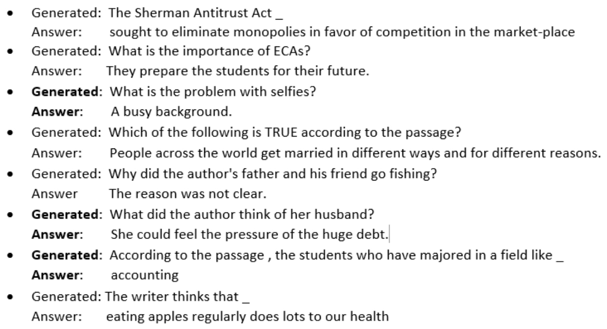
Table 1 summarises the results of our question generation models[10]. We can see that the GRU-based model has the worst performance.

The BERT based model outperforms RNN and achieves the performance metric that is twice (in some, cases three times) more. It is also the only one model that achieves non-zero score on BLEU-4 metric. We can assume that in some cases BERT is able to generate long phrases that correspond to the ones in target.

T5 model achieves even better results on every metrics (compared to the previous two models). We can see the increase over all metrics. The model improves on generation of n-grams and it also seem to generate more tokens that are closer to the ones in the target (see ROUGE-L).

The addition of QJ brings only a small increase in BLEU-1, BLEU-2 and ROUGE-L. It is not clear whether the increase is explained by the number of sequences that we generate per sample or by the filtering mechanism. In our experiments we generate 3 sequences per sample to evaluate T5 model and 7 for T5-QJ. The T5-QJ chooses questions with the quality above *0.97*, so the amount of generated questions per target varies. In Section 4.2.1 we show and discuss the output of the judge model.

The **output of the T5-Model** is displayed on Figure 2. The generated questions are mostly consistent with the answer. It does backfire in several cases, for example, when provided with the answer *She could feel the pressure of her husband*, the model generates the question that does not correspond well to the answer, i.e. What did the author think of her husband. Our assumption was that model would *over-generate* questions that are mostly phrased as *In this article we learnt* _, however, the model seems to generate diverse type of questions.



- Generated: The Sherman Antitrust Act _
  Answer: sought to eliminate monopolies in favor of competition in the market-place
- Generated: What is the importance of ECAs?
  Answer: They prepare the students for their future.
- **Generated**: What is the problem with selfies?
  Answer: A busy background.
- Generated: Which of the following is TRUE according to the passage?
  Answer: People across the world get married in different ways and for different reasons.
- Generated: Why did the author's father and his friend go fishing?
  Answer The reason was not clear.
- **Generated**: What did the author think of her husband?
  Answer: She could feel the pressure of the huge debt.
- **Generated**: According to the passage , the students who have majored in a field like _
  Answer: accounting
- Generated: The writer thinks that _
  Answer: eating apples regularly does lots to our health

Figure 2: Output of the T5 Question generator. Each bullet point consist of a generated question and corresponding correct answer (ground truth). The highlighted pairs identify *incoherentpoor* pairs. The underscore line stands for *blank*

4.2 Evaluation of distractor generators

Table 2 shows the result for distractor generation of the three model. As shown from the table, although BERT-based model contains a significant larger amount of parameters compared to the Seq2Seq model, its performance metrics is only slightly better than that of the Seq2Seq model. This is an evidence showing that BERT-based model can be further finetuned and trained to exploit it capacity.

T5 model outperforms GRU-based and BERT-based model in BLEU-1, METEOR, and ROUGE-L metrics. This can imply that the tokens in the distractors generated from T5 mostly appear in the target distractor. High ROUGE-L score indicates that the T5's generated distrators have close representation with that of the target distractor. Zero scores in BLEU-2, BLEU-3, and BLEU-4 indicate that the generated tokens of T5 almost do not appear with the same order as in the target sequences.

Evaluation based on BLUE, METEOR and ROUGE-L might not be the best way to benchmark the T5-distractor generator. During training our targets are random samples from the associated distractor set. We ask our model to generate the random distractors that are consistent with text, but do not resemble the answer. Thus, we evaluate random target versus randomly generated distractor. Even with that in mind, the performance of T5-distractor generator seem to significant increase on ROGUE-L.

The output produced by the T5-Distractor Generator is shown on Figure 3. We can see that model identifies what kind of distractor it should generate, e.g. it is able to see when it should generate names or websites.

In some cases the distractor is just a random fact picked up from the passage and not related

[9] Questions on the form *In this article, we learnt* _

[10] T5-QJ stands for T5 model with the Question Judge unit

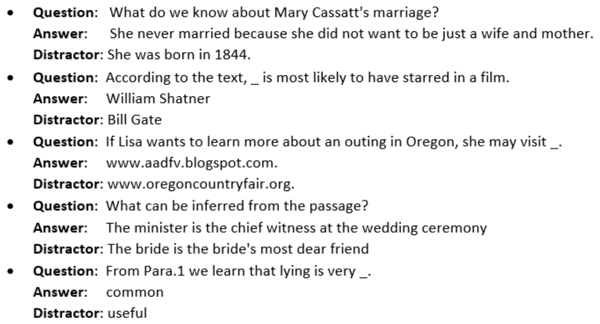
Table 2: Experimental results of our models on distractor generation

	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L
Seq2Seq	0.08	0.02	0.01	0.00	0.10	0.16
BERT	0.14	0.04	0.02	0.01	0.14	0.18
T5	0.40	0.00	0.00	0.00	0.33	0.83

to the content of questions. For example, in case of `What do we know about Mary Cassatt's marriage?`, the distractor is a piece of information about Mary and not her marriage.

In the second question, the distractor is a randomly picked up name (from the passage), this name appears in a totally different context.

In case of more general questions like `What can be inferred from the passage?` the model seem to generate gibberish. `The bride is the bride's most dear friend` - which does not make much sense (plus, this *idea* does not appear in the article).

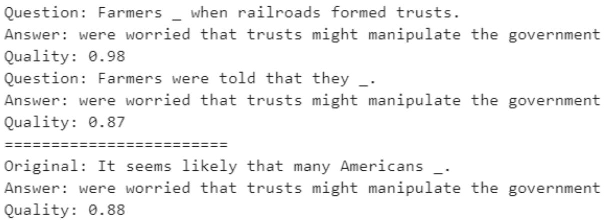


- **Question**: What do we know about Mary Cassatt's marriage?
  Answer: She never married because she did not want to be just a wife and mother.
  Distractor: She was born in 1844.
- **Question**: According to the text, _ is most likely to have starred in a film.
  Answer: William Shatner
  Distractor: Bill Gate
- **Question**: If Lisa wants to learn more about an outing in Oregon, she may visit _.
  Answer: www.aadfv.blogspot.com.
  Distractor: www.oregoncountryfair.org.
- **Question**: What can be inferred from the passage?
  Answer: The minister is the chief witness at the wedding ceremony
  Distractor: The bride is the bride's most dear friend
- **Question**: From Para.1 we learn that lying is very _.
  Answer: common
  Distractor: useful

Figure 3: Output of the `T5`-Distractor Generator. You can see ground-truth questions, correct answer and **generated** distractors

4.2.1 Output of the Judge model

Figure 4 shows the example an output produced by `T5 + QJ` model. In case of this example, `QJ` successfully chooses a more coherent pair (with the Quality of 0.98).



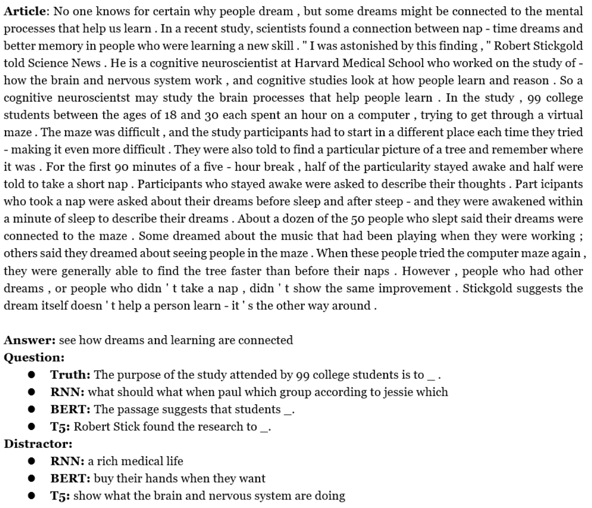
```
Question: Farmers _ when railroads formed trusts.
Answer: were worried that trusts might manipulate the government
Quality: 0.98
Question: Farmers were told that they _.
Answer: were worried that trusts might manipulate the government
Quality: 0.87
=======================
Original: It seems likely that many Americans _.
Answer: were worried that trusts might manipulate the government
Quality: 0.88
```

Figure 4: Output of the `T5+QJ` model. The first two questions are generated by the T5-model. The answer is a correct ground truth answer. Quality is calculated by the Judge model.

However, we can see that second pair also receives high score, even thought it does not make a perfect sense (even if you do not know the content of the article - you can clearly see that). Thus, it confirms the description given in the document: the `QJ` model does not look at the correctness of answer. Another interesting observation is related to the ground-truth question. On Figure 4, the combination of the ground-truth question and the ground-truth correct answer does not receive the highest score. We might assume that the score value is higher in the first pair because both sequences (i.e. generated question and answer) both share same word *trusts*. However, further investigation is needed.

5 Discussion and Conclusion

We performed experiments on `RNN`, `BERT`, and `T5` proved that it is possible to build an end-to-end system generating multiple-choice questions given an article and an answer.



Article: No one knows for certain why people dream , but some dreams might be connected to the mental processes that help us learn . In a recent study, scientists found a connection between nap - time dreams and better memory in people who were learning a new skill . " I was astonished by this finding , " Robert Stickgold told Science News . He is a cognitive neuroscientist at Harvard Medical School who worked on the study of - how the brain and nervous system work , and cognitive studies look at how people learn and reason . So a cognitive neuroscientst may study the brain processes that help people learn . In the study , 99 college students between the ages of 18 and 30 each spent an hour on a computer , trying to get through a virtual maze . The maze was difficult , and the study participants had to start in a different place each time they tried - making it even more difficult . They were also told to find a particular picture of a tree and remember where it was . For the first 90 minutes of a five - hour break , half of the particularity stayed awake and half were told to take a short nap . Participants who stayed awake were asked to describe their thoughts . Part icipants who took a nap were asked about their dreams before sleep and after steep - and they were awakened within a minute of sleep to describe their dreams . About a dozen of the 50 people who slept said their dreams were connected to the maze . Some dreamed about the music that had been playing when they were working ; others said they dreamed about seeing people in the maze . When these people tried the computer maze again , they were generally able to find the tree faster than before their naps . However , people who had other dreams , or people who didn ' t take a nap , didn ' t show the same improvement . Stickgold suggests the dream itself doesn ' t help a person learn - it ' s the other way around .

Answer: see how dreams and learning are connected
Question:
- **Truth:** The purpose of the study attended by 99 college students is to _ .
- **RNN:** what should what when paul which group according to jessie which
- **BERT:** The passage suggests that students _.
- **T5:** Robert Stick found the research to _.

Distractor:
- **RNN:** a rich medical life
- **BERT:** buy their hands when they want
- **T5:** show what the brain and nervous system are doing

Figure 5: An example generations of our three models

However, the quality depends on the choices of models. Figure 5 shows question and distractors generated from our models based on an article about research on sleeping. Without proper semantic knowledge, `RNN` is struggling to generate a coherent sentence. `BERT` model knows the correct method to organise a sentence, but it cannot understand the relation between the answer and the article. As a result, it cannot ask a proper question related to the answer. `T5` model did a good job in both asking questions and generating distractors. The `GRU`-based model contains components that are trained from randomly initialised parameters. At the same time, BERT has only one decoder layer, while the T5 model is a pre-trained model for text generation. This could be an indicator that pre-training language models with a flexible decoder are beneficial for downstream tasks.

6 Devision of labor

Haonan: I was responsible for the implementation of the Seq2Seq generation model and the training of Seq2Seq and BERT-based model. I implemented the pipeline for the generation and display of sentences and top-p sampling method for models that are not from Huggingface. I also contributed to the integration of our code. For our project documents, I wrote the description of Seq2Seq model and contributed to the discussion about the comparison of our three models.

Vinh: My main contribution is on implementing data pre-processing scripts and the pipeline for preparing and serving data to the models. Furthermore, I implement the pipeline for computing the performance metrics given the model, as well as module to structure the model into convenient configuration files. I also contribute to conduct code quality, refactoring, and management. For model development, I am responsible for implementing BERT-based model. In term of project documents, I contribute mostly to the project plan, explanation of BERT-based model, as well as discussion on performance metrics table 2.

Germans: I was responsible for the implementation of the T5 Question Generation model, Question Judge and T5-distractor generation model. I implemented and run the hyperparameter search for RNN-model and Top-K/Top-p sampling. Meanwhile, I run the testing for the `T5`, `T5+QJ` model. In the earlier stages I contributed to the Exploratory Data Analysis. In terms of the final report, I contributed to the Introduction, Methods, Experiments, Results and Conclusion chapters. To be more precise, I described the theory and setup of `T5`-related models, as well as theory behind sampling and tuning. I also contributed to the explanation of metrics.

References

Bergstra, J., Yamins, D. & Cox, D. (2013), Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures, *in* 'International conference on machine learning', PMLR, pp. 115–123.

Callison-Burch, C., Osborne, M. & Koehn, P. (2006), Re-evaluating the role of bleu in machine translation research, *in* '11th Conference of the European Chapter of the Association for Computational Linguistics'.

Chan, Y.-H. & Fan, Y.-C. (2019), A recurrent bert-based model for question generation, *in* 'Proceedings of the 2nd Workshop on Machine Reading for Question Answering', pp. 154–162.

Denkowski, M. & Lavie, A. (2014), Meteor universal: Language specific translation evaluation for any target language, *in* 'Proceedings of the ninth workshop on statistical machine translation', pp. 376–380.

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2018), 'Bert: Pre-training of deep bidirectional transformers for language understanding', *arXiv preprint arXiv:1810.04805* .

Fan, A., Lewis, M. & Dauphin, Y. (2018), 'Hierarchical neural story generation', *arXiv preprint arXiv:1805.04833* .

Ganesan, K. (2018), 'Rouge 2.0: Updated and improved measures for evaluation of summarization tasks', *arXiv preprint arXiv:1803.01937* .

Gao, Y., Bing, L., Li, P., King, I. & Lyu, M. R. (2019), Generating distractors for reading comprehension questions from real examinations, *in* 'Proceedings of the AAAI Conference on Artificial Intelligence', Vol. 33, pp. 6423–6430.

Goodfellow, I., Bengio, Y. & Courville, A. (2016), *Deep Learning*, MIT Press. `http://www.deeplearningbook.org`.

Holtzman, A., Buys, J., Du, L., Forbes, M. & Choi, Y. (2020), 'The curious case of neural text degeneration'.

Jia, X., Zhou, W., Sun, X. & Wu, Y. (2020), 'Eqg-race: Examination-type question generation', *arXiv preprint arXiv:2012.06106* .

Jurafsky, D. & Martin, J. H. (2020), 'Speech and language processing (3rd edition, draft)', *Chapter 11: Machine Translation and Encoder-Decoder Models. Retrieved April 2021* .

Lai, G., Xie, Q., Liu, H., Yang, Y. & Hovy, E. (2017), 'Race: Large-scale reading comprehension dataset from examinations', *arXiv preprint arXiv:1704.04683* .

Lelkes, A. D., Tran, V. Q. & Yu, C. (2021), 'Quiz-style question generation for news stories', *arXiv preprint arXiv:2102.09094* .

Li, L., Jamieson, K., Rostamizadeh, A., Gonina, E., Hardt, M., Recht, B. & Talwalkar, A. (2018), 'A system for massively parallel hyperparameter tuning', *arXiv preprint arXiv:1810.05934* .

Liaw, R., Liang, E., Nishihara, R., Moritz, P., Gonzalez, J. E. & Stoica, I. (2018), 'Tune: A research platform for distributed model selection and training', *arXiv preprint arXiv:1807.05118* .

Liu, B., Wei, H., Niu, D., Chen, H. & He, Y. (2020), Asking questions the human way: Scalable question-answer generation from text corpus, *in* 'Proceedings of The Web Conference 2020', pp. 2032–2043.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. & Liu, P. J. (2019), 'Exploring the limits of transfer learning with a unified text-to-text transformer', *arXiv preprint arXiv:1910.10683* .

Scialom, T., Piwowarski, B. & Staiano, J. (2019), Self-attention architectures for answer-agnostic neural question generation, *in* 'Proceedings of the 57th annual meeting of the Association for Computational Linguistics', pp. 6027–6032.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. & Polosukhin, I. (2017), 'Attention is all you need', *arXiv preprint arXiv:1706.03762* .

Zhao, Y., Ni, X., Ding, Y. & Ke, Q. (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', pp. 3901–3910.