# EECS4088: FINAL PROJECT REPORT (Dataset Generation, and Evaluation for Natural Question Generation)

Amirhossein Razavi
ahr91@my.yorku.ca
216715963

## ABSTRACT

Natural Language Processing (NLP) is one of the Artificial Intelligence branches that analyzes and understands language as humans do. NLP has many applications such as Information Retrievals, Information Extraction, Question-Answering, Machine Translation, etc. This project is an NLP project as it needs the machine to understand a sentence like a human and generate a natural-sounding question based on its understanding. This project aims to Create a Question-Answering dataset that contains natural questions and a single sentence as the answer to each question. We emphasize having natural questions so the resulting dataset would be practical to use in the question generation tasks in the real world. We will then test the sentences of the resulting dataset with BART and T5 models to generate their corresponding questions. After generating the questions, We will evaluate the quality of the dataset and the performance of the models used to create this dataset based on both automatic and human evaluation.

## 1 INTRODUCTION

Question generation is a task in NLP which concerns how to generate a question from a text document. This project aims to generate a sentence-based QA dataset that can be used as training data in the real world's NQG (Natural Question Generation) tasks. The questions generated using a system trained by this dataset should be like the questions that a human would ask (that is why we call them natural questions). To achieve this goal, the questions in the dataset itself need to be natural, too, as it gives the ability to think like a human (when generating a question) to the system. Also, the sentences in the generated dataset should be definite and complete so, "good" questions can be generated by these sentences.

Question generation can have many applications, such as in education, reading comprehension, and developing a knowledge base for conversational systems.

The most effective methods for the question generation task are to use neural sequence-to-sequence methods. These methods need a large set of labeled training data (Question-Answer pairs). There are many existing datasets like Google's Natural Question Dataset, such as SQuAD, but the downside of them is that the questions in them are not natural, meaning that the questions were not humanly generated, and they are obtained by looking at the answers through; crowdsourcing for example. There are other datasets too that have natural questions, such as NewsQA, but they also have another downside which is that they have a paragraph as their long answer, which has lots of information, and many questions can be generated from that one paragraph alone. They also have a word or phrase long short answer, which does not mean anything special to the system.

In these datasets, sentence-based natural Question Answering datasets are lacking, and some neural question generation models can only take a single sentence as an answer to generate questions which would mean that by having the mentioned datasets, there would not be an easy way to use some of neural question generation models.

After generating the desired dataset, we will evaluate two popular and well-respected neural question generation methods, BART and T5, using the generated dataset.

To conclude, the objectives of this project are as follows:

(1) Create a Question-Answering dataset that contains natural questions, and a single sentence as the answer to each question. We emphasize containing natural questions so the resulting dataset would be practical to use in the question generation tasks in the real world.
(2) Evaluate two state-of-the-art neural question generation methods. BART and T5.

## 2 METHODOLOGY FOR DATASET GENERATION

### 2.1 Original Dataset

To help the development of this natural question generation task, we use the Natural Questions (NQ) corpus provided by Google. The dataset was initially designed to help spur development in open-domain question answering. We will use this dataset for the question generation task in this project as in Google's Natural Questions Dataset; the questions consist of real anonymized, aggregated queries issued to the Google search engine. Simple heuristics are used to filter questions from the query stream. Thus the questions are "natural" because they represent actual queries from people seeking information [5].

The dataset contains (question, Wikipedia page, long answer, short answer) quadruples where: the question seeks factual information; which the Wikipedia page may or may not have the information required to answer the question; the long answer is a bounding box on this page containing all information necessary to infer the answer, and the short answer is one or more entities that give a short answer to the question, or a boolean yes or no. The long and short answers can be NULL if no viable candidates exist on the Wikipedia page.

By having a massive amount of data, natural questions, and a broad range of topics, this dataset makes an excellent match for our question generation task [4]. The number of data in the original dataset is shown in Table 1.

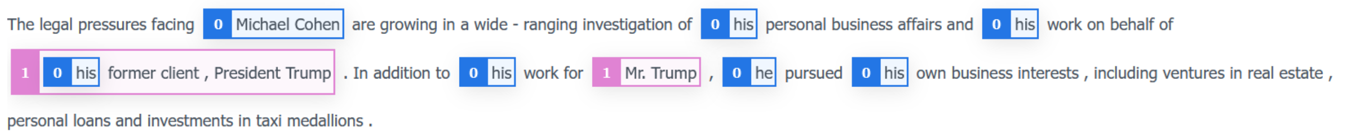
The legal pressures facing 0 Michael Cohen are growing in a wide - ranging investigation of 0 his personal business affairs and 0 his work on behalf of 1 0 his former client , President Trump . In addition to 0 his work for 1 Mr. Trump , 0 he pursued 0 his own business interests , including ventures in real estate , personal loans and investments in taxi medallions .

**Figure 1: Coreference Resolution Example.**

**Table 1: Google's Natural Questions Dataset**

| Dataset | Number of data |
|---|---|
| Google's Natural Questions Dataset | 307373 |

## 2.2 Data Parsing

Not all the data in the original dataset was useful for the question generation task. Thus, we had to filter out the useless data. We achieved this goal by parsing the data to make it ready for training the system. The dataset was in the format of JSONL, where each line represented a dictionary consisting of 4 keys: question, Wikipedia page, long answer, and short answer. The following steps are how the data parsing was done, we chose the data that:

(1) Had both short answer and long answer. As we wanted to be able to extract the sentence of short answer and the only way to be able to do that is to have both short answer and long answer together in a data.
(2) Had the short answer existed in the long answer. As in some cases in the dataset the short answer did not exist in the long answer and in order to find the sentence of short answer, we need it to be in long answer.
(3) Had the sentence of short answer being less than 50 words. As we want the system to be able to predict a question based on a sentence, having more than 50 words in that sentence would confuse the system as it can generate multiple questions regarding that one sentence. The average word count in a sentence is 15 to 20 words.
(4) Did not contain any kind of HTML tag (list, table, etc.) in its short answer. As we want to be able to extract the sentence of short answer from the long answer, there must be a sentence existing. Having the short answer in lists, tables or any other form that is not sentence based would not help our development of the project and that is why we filtered the data that its answer was in format of list or table.
(5) The corresponding question starts with a "wh" question word. As in the original dataset there were lots of queries that did not look like a question, and they were more like a sentence like "number of red cards in world cup 2018". We want the system to perform at its best and ideally create grammatically correct questions and these types of data would just confuse the system about what the correct way of asking a question is.
(6) Had the "`sentence`" of short answer contain only one sentence. As there were some cases that the question had multiple short answers and the short answers were not in the same sentence which would cause our "`sentence`" of short answer to consist more than one sentence. We also want to filter out them too as the goal is for the system to be able to generate a question based on one sentence.
(7) If the current data that the system is checking passes all of these conditions, we would have it in our files.

Here is an examples of the parsed data:

**Question** WHAT YEAR DID THE MOVIE DEUCES COME OUT

**Long answer** DEUCES IS AN AMERICAN CRIME DRAMA WRITTEN AND DIRECTED BY JAMAL HILL . THE FILM STARS LARENZ TATE , MEAGAN GOOD , LANCE GROSS AND SIYA . THE FILM IS EXECUTIVE PRODUCED BY QUEEN LATIFAH FOR HER PRODUCTION COMPANY FLAVOR UNIT ENTERTAINMENT . DEUCES PREMIERED ON NETFLIX ON APRIL 1 , 2017 .

**Short answer from the original dataset** 2017

**Short answer from data parsing** DEUCES PREMIERED ON NETFLIX ON APRIL 1 , 2017

As we can see in the example, the original short answer was "2017" which we cannot use to generate the question of "what year did the movie deuces come out". When the data is parsed the way described, the short answer changes to be "Deuces premiered on NETFLIX on April 1, 2017" which makes it possible for the system to generate that question based on this sentence alone.

Let us name the generated dataset using data parsing to be "`Sentence-based Natural Question Dataset`" or in short, "`SNQD`". Number of data in SNQD is shown in Table 2.

**Table 2: Sentence-based Natural Question Dataset or SNQD**

| Dataset | Number of data |
|---|---|
| SNQD | 61374 |

## 2.3 Coreference Resolution

Coreference resolution is the task of finding all expressions that refer to the same entity in a text. As the example in Figure 1 shows that the marked words with the blue color refer to "Michael Cohen", and the pink color refers to "Michael Cohen's client, president Trump". Let's say the second sentence in this example is extracted, and we want to generate a question based on it. We cannot develop a good question as we do not have the "Michael Cohen" name in the second sentence. So, the idea of resolving the coreference resolution is to find the referred entities and replace them with their root. Resolving or identifying such coreferences is essential for NLP tasks involving natural language understanding, such as information extraction and question generation.

To find these coreference resolutions, AllenNLP's library [3] was used, which returns the clusters of the representation of each span in the document. Then we replaced each with the referred entity. In the replacing process, we found M. Maślankowska's article helpful as it had some extra features added to the original resolving implementation by AllenNLP library [8].

The procedure for finding and editing the short answers in the SNQD dataset is as follows:

(1) By giving the long answers in the original dataset to the system, it would find its coreferences and return the clusters of coreferences in the long answers.
(2) Each cluster is checked to see if it contains any referring entity in the short answer. The referring entities not in the short answer are deleted since we need the short answer's coreferences to be resolved. If there is no referring entity in the short answer, the short answer itself is returned, meaning that there was no coreference in our short answer.
(3) We remove the clusters with only "Pronouns" in them. As there is no noun in the cluster, we cannot replace the pronouns with their referring noun. This kind of issue will result in a meaningless sentence such as "I will pick I up after I work is done" instead of "I will pick her up after her work is done" (if the cluster chooses "I" as the head and "her" as its referring entity). The Spacy library is used to help with detecting the roles of the words [8].
(4) AllenNLP detects cataphora, but since it treats the first mention in a cluster as its head, it leads to other errors. The cataphor (e.g., a pronoun) precedes the postcedent (e.g., a noun phrase), so a meaningless span becomes a cluster's head. To avoid this issue, we take the first noun in the cluster as its head so that a pronoun cannot act as a head in any of the clusters [8].
(5) There is a possibility of having multiple coreferences on the same span. The issue with this event is that the order of resolving the coreferences will change the final sentence that we would have. In the case of this event, we take the most inner span and resolve that one only to overcome this issue [8].

Although this methodology solved the coreferences, we found some indexing issues due to the different indexing approaches used by Google's Natural Question Dataset and the AllenNLP's coreference resolving method. This difference caused sentences in some of the data to contain a part of the previous sentence, too, which would make the sentence not understandable. We removed these sentences as they were incomplete and not informative. The number of data that had this issue was 1618.

As all the coreferences have been solved in the new dataset and are different from the previously generated dataset by using data parsing, let us call it "Coref Resolved SNQD". The number of data in Coref Resolved SNQD is shown in Table 3.

**Table 3: Coreference Resolved SNQD**

| Dataset | Number of data |
|---|---|
| Coref Resolved SNQD | 59756 |

## 3 METHODOLOGY FOR QUESTION GENERATION

The version of question generation used in this project is taken from Suraj Patil's GitHub repository [10].

### 3.1 Sequence-to-Sequence Models

The sequence-to-sequence models take a sequence as input and output another sequence like when we use Google Translate, we give it a sentence in Spanish, for example, and it returns a sentence in English. These sequences can be of many forms, such as paragraphs, sentences, words, letters, time series, etc. Question Generation task is a sequence-to-sequence task since it takes a sentence, for example, as input and outputs a question which is a sequence of words [13].

### 3.2 BART Algorithm

BART is a denoising autoencoder for pretraining sequence-to-sequence models. BART is trained by (1) corrupting text with an arbitrary noising function, and (2) learning a model to reconstruct the original text. It uses a standard Transformer-based neural machine translation architecture which, despite its simplicity, can be seen as generalizing BERT (due to the bidirectional encoder), GPT (with the left-to-right decoder), and many other more recent pretraining schemes [6].

### 3.3 T5 Algorithm

T5 stands for Text-To-Text Transfer Transformer. T5 is an encoder-decoder model pre-trained on a multi-task mixture of unsupervised and supervised tasks and for which each task is converted into a text-to-text format. T5 is trained using teacher forcing, which means that for training, we always need an input sequence and a corresponding target sequence. The T5 model works well on various tasks without any significant modifications to the model. T5 is one of the well-known algorithms for the question generation task which we tested in this project [11].

## 4 EVALUATION

### 4.1 Experiment Setup

Since the dataset has many data and the BART algorithm is vast, the experiment process will require too much memory and CPU. This experiment has to be done on online servers such as Google Cloud or the Brain Servers to avoid the memory or CPU outages and allow us to use high-performance GPUs for the computations. The experiment was done on Google Cloud servers, on a deep learning GCE (Google Compute Engine). The machine type used was n1-standard-8 (8 vCPUs, 30 GB memory), with 1 NVIDIA Tesla K80. Python version 3.9.6 is used to calculate and manage the result data from the dataset.

### 4.2 Automatic Evaluation

Automatic evaluation means comparing specific models or datasets based on some generated scores. In the field of NLP, there are many types of these scorings, but we chose three of the most related ones to our project, $BLEU$, $METEOR$, and $ROUGE_L$ Scores.

The automatic evaluation in this project was done using "nlg-eval" software implemented by S. Sharma [12].

Below is a brief explanation of each of these scores:

*BLEU* or Bilingual Evaluation Understudy, was first introduced for the machine translation task. Its objective is to reduce human labor's required time and effort in evaluating the translations. It is also language-independent, inexpensive, quick, and highly correlated with human judgment. *BLEU* measures how similar a candidate text sequence is to a reference text sequence. It is based on n-gram matching, where the n can be from 1 to 4 [1, 9].

*METEOR* or Metric for Evaluation of Translation with Explicit ORdering, is based on the harmonic mean of unigram precision and recall, with weighting recall higher than precision. It cores predictions by aligning them to ground truth sentences and measuring the sentence-level similarity scores. It performs these alignments with the help of exact word matching, stemming, synonyms, and paraphrases [1, 2].

*ROUGE* or Recall-Oriented Understudy for Gisting Evaluation, was proposed for evaluating text summarization and machine translation. There are a few variations of the ROUGE metric. The one we are using for this project is $ROUGE_L$, which compares the generated summary or translation with the references based on the Longest Common Subsequence (LCS) [1, 7].

## 4.3 Test-Run (Car Manual Dataset)

To check for the effectiveness of our evaluation process, we ran a test-run on a dataset called Car Manual Dataset. This dataset has Question Answer pairings from a car manual where the answers are mostly single sentences. The size of the dataset is given in the Table 4.

**Table 4: Car Manual Dataset or CMD**

| File | Train-set size | Dev-set size | Test-set size |
|---|---|---|---|
| Car Manual Dataset | 4360 | 360 | 312 |

This dataset was great to test our evaluation process as the sentences are complete and the size of the test set is not large. In Table 5, we can see the scores for automatic evaluation on the Car Manual dataset. Both of our models (BART and T5) did well, but BART did better than T5 in all cases.

**Table 5: Metrics of running BART and T5 Models on Car Manual dataset**

| Dataset | Model | $BLEU1$ | $BLEU2$ | $METEOR$ | $ROUGE_L$ |
|---|---|---|---|---|---|
| CMD | BART | 0.560175 | 0.498418 | 0.391080 | 0.576190 |
| | T5 | 0.548173 | 0.478553 | 0.370160 | 0.544810 |

In the human evaluation part, we checked the questions generated and compared them with the targeted questions. The generated questions were correct in most cases, and they had little to no difference from the targeted questions. The differences were mainly grammatical, with both having correct grammar. The T5 did slightly better in the human evaluation, as we can see in the below cases:

**Case 1:**

**Sentence** DO NOT CLEAN OR WIPE LAMP COVERS WHEN DRY .

**BART generated question** SHOULD I CLEAN LAMP COVERS WHEN DRY?

**T5 generated question** CAN I WIPE LAMP COVERS WHEN DRY?

**Targeted question** CAN I CLEAN LAMP COVERS WHEN DRY ?

**Case 2:**

**Sentence** THE LATCH ATTACHMENTS ON THE CHILD RESTRAINT ARE USED TO ATTACH THE CHILD RESTRAINT TO THE ANCHORS IN THE VEHICLE .

**BART generated question** WHAT ARE THE LATCH ATTACHMENTS MADE OF?

**T5 generated question** WHAT ARE THE LATCH ATTACHMENTS ON THE CHILD RESTRAINT?

**Targeted question** WHAT DO THE LATCH ATTACHMENTS ON THE CHILD RESTRAINT DO ?

**Case 3:**

**Sentence** TO PREVENT DAMAGE ALWAYS FOLLOW THESE CLEANING INSTRUCTIONS. BE SURE THE MOLDING IS COOL TO THE TOUCH BEFORE APPLYING ANY CLEANING SOLUTION. USE ONLY APPROVED CLEANING SOLUTIONS FOR ALUMINUM , CHROME , OR STAINLESS STEEL. SOME CLEANERS ARE HIGHLY ACIDIC OR CONTAIN ALKALINE SUBSTANCES AND CAN DAMAGE THE MOLDINGS. USE A NONABRASIVE WAS ON THE VEHICLE AFTER WASHING TO PROTECT AND EXTEND THE MOLDING FINISH .

**BART generated question** HOW CAN I PREVENT DAMAGE WHILE CLEANING THE INTERIOR CARPETS?

**T5 generated question** WHAT SHOULD I DO TO PREVENT DAMAGE FROM THE MOLDINGS?

**Targeted question** HOW CAN I PREVENT DAMAGE TO THE BRIGHT METAL MOLDINGS ?

In cases number 1 and 2, both of the algorithms did well, with T5 doing better than BART, but in the case number 3, which had multiple sentences in itself, the models performed very poorly, and this is why the SNQD dataset (our newly generated dataset) consists only answers with a single sentence. Thus, we will not be facing this issue in our main tests.

## 4.4 Data Parsing Evaluation

When we did the data parsing by the method explained in the Methodology For Dataset Generation part of the report, the final dataset had the following attributes.

(1) There were two files generated by this script, one for questions and one for modified short answers.

(2) Each line in either file represents the same line in the other file.
(3) Modified short answers consist of only one sentence, and the maximum word count for it is 50 words.
(4) There are no tables, lists, or list items in modified short answers as they would be unrelated for our project"s purposes.
(5) Only the data were chosen that had both short answer and long answer so we could obtain the sentence that the short answer exists in long answer, and the modified short answers file is basically a file containing these sentences.

We needed to evaluate it manually as there are no good ways to assess it except by looking at random lines and checking if they look rational or not. The way we did the data parsing, the results were entirely reasonable, but the trade-off made was the original dataset had 307,373 rows, and our parsed data had only 61,374 rows which are around 20% of the original dataset.

## 4.5 Coreference Resolution Evaluation

*4.5.1 Without Implementation of Coreference Resolution.* When we did the data parsing, we noticed that many sentences referred to the words before that sentence in the long answer, like having pronouns such as he, she, they, their, etc. This made the sentences in SNQD "incomplete" which means that the sentences did not have the complete information for the right question to be generated.

Take this sentence for example:

**Sentence from SNQD** THE NAME IS DERIVED FROM PATRONYMIC FORM OF THE NAME HENDRY , WHICH IS A SCOTTISH FORM OF HENRY

**Question from SNQD** WHERE DOES THE HENDERSON LAST NAME COME FROM

As we can see, no one can generate a valid natural question based on this sentence alone since the information is incomplete (what is "the name"?). Also, the original question asks for the "Henderson" last name origins, and there is no "Henderson" in the sentence given to the system.

As the number of information changes with the number of words in SNQD, we wanted to test if changing the number of words in SNQD would change its performance as well or not, so we used SNQD and three modified versions of SNQD with having 0-30, 10-30, 30-50 words each of their sentences. As we can see in Table 6, the scores are much lower than the scores we had for Car Manual Dataset, in any case. The SNQD 30-50 had the highest scores among the rest. So by changing the number of words, the scores changed, but they were not even close to what we had for the Car Manual dataset.

**Table 6: Metrics of running BART and T5 Models on SNQD and its modified versions**

| Dataset | Model | $BLEU1$ | $BLEU2$ | $METEOR$ | $ROUGE_L$ |
|---|---|---|---|---|---|
| SNQD (0-50) | BART | 0.323440 | 0.225199 | 0.166647 | 0.300929 |
| SNQD (0-30) | BART | 0.366399 | 0.263268 | 0.188732 | 0.354553 |
| SNQD (10-30) | BART | 0.346400 | 0.242911 | 0.179168 | 0.322215 |
| SNQD (30-50) | BART | 0.376816 | 0.276793 | 0.198042 | 0.375410 |

*4.5.2 With Implementation of Coreference Resolution.* After implementing the coreference resolution, the sentences became complete, and the system could generate natural questions based on these sentences. By resolving the coreference in the example above, the sentence would change to:

**Sentence from Coref Resolved SNQD** HENDERSON IS DERIVED FROM PATRONYMIC FORM OF THE NAME HENDRY , WHICH IS A SCOTTISH FORM OF HENRY

**Question from Coref Resolved SNQD** WHERE DOES THE HENDERSON LAST NAME COME FROM

As we can see, a coreference resolved sentence is a complete sentence, and the system should be able to predict its corresponding question. Since all the coreferences have been resolved in SNQD, the number of words in each sentence has changed, but it does not matter since the sentences are now complete, and we do not need to run tests on different counts of words. The performance scores for the coreference resolved SNQD are as shown in Table 7. We can see that the scores are much closer to the scores we had in the Car Manual Dataset.

**Table 7: Metrics of running BART and T5 Models on Coref Resolved SNQD**

| Dataset | Model | $BLEU1$ | $BLEU2$ | $METEOR$ | $ROUGE_L$ |
|---|---|---|---|---|---|
| Coref Resolved SNQD | BART | 0.511272 | 0.391012 | 0.261638 | 0.498220 |
| Coref Resolved SNQD | T5 | 0.464178 | 0.342072 | 0.227008 | 0.452510 |

*4.5.3 Comparing With/Without Implementation of Coreference Resolution.* Below is an example comparing the questions generated by BART and T5, with or without implementing coreference resolution in SNQD. Without the implementation of coreference resolution, the sentence does not contain any word of "Dynasty", and it mentions it as "The series", but with the implementation of coreference resolution, the name of "Dynasty" is mentioned. The generated questions from BART and T5 before resolving coreferences are not correct. The BART algorithm even says the name of another series which has nothing to do with "Dynasty". After resolving the coreferences, both the BART and T5 generated correct questions, even better than the targeted question itself.

**Targeted question** WHEN DOES THE NEXT EPISODE OF DYNASTY COME OUT

**Without implementation of coreference resolution:**

**Sentence from SNQD** ON APRIL 2 , 2018 , THE CW RENEWED THE SERIES FOR A SECOND SEASON , WHICH IS SET TO PREMIERE ON OCTOBER 12 , 2018

**BART generated question** WHEN DOES THE NEW EPISODE OF RIVERDALE COME OUT

**T5 generated question** WHEN DOES THE NEW SEASON OF THE TV SHOW COME OUT

**With implementation of coreference resolution:**

**Sentence from Coref Resolved SNQD** ON APRIL 2 , 2018 , THE CW RENEWED DYNASTY FOR A SECOND SEASON , WHICH IS SET TO PREMIERE ON OCTOBER 12 , 2018

**BART generated question** WHEN DOES DYNASTY COME BACK FOR SEASON 2

**T5 generated question** WHEN DOES THE NEW SEASON OF DYNASTY START

## 4.6 Duplicate Short Answers

Another issue that we faced while working on the project is that the original dataset contained duplicate short answers meaning there could be multiple questions corresponding to the exact text. As a result, our newly generated datasets (SNQD and Coref Resolved SNQD) also have those duplicates. This event has its benefits and drawbacks, of course.

One of the benefits is it gives the system chance to understand the texts in multiple ways. We, humans, can generate various questions from a sentence (depending on the sentence), and by having duplicates in the dataset, we give that chance to the system.

One of the drawbacks is that there is a possibility of having the same sentence in the training and testing dataset, leading to overfitting.

To further understand this issue, we took three approaches:

**Coref Resolved SNQD (Dupes)** Allowing duplicates. It would be the same dataset (Coref Resolved SNQD) as we have currently.

**Unique CRSNQD** Only having unique sentences present in the dataset.

**Moderate CRSNQD** Keeping up to a maximum of three questions per sentence.

The sizes of these three modified versions of the Coref Resolved SNQD are as shown below, Table 8:

**Table 8: Coreference Resolved SNQD**

| Dataset | Number of data |
|---|---|
| Coref Resolved SNQD (Dupes) | 59756 |
| Unique CRSNQD | 45605 |
| Moderate CRSNQD | 55863 |

As we tested the three approaches, the Coref Resolved SNQD had the best, and the Unique CRSNQD dataset had the worst scores, but that might be due to having more understanding of the data or overfitting. If we look at the following Table 9, T5 scores do not change much between the different datasets, but the BART model's scores change dramatically. In the Coref Resolved SNQD, BART and T5 have almost 0.05 difference between their scores, but in the unique CRSNQD, their difference becomes down to 0.01, but still, the BART did better than T5.

**Table 9: Metrics of running BART and T5 Models on SNQD and its modified versions**

| Dataset | Model | $BLEU1$ | $BLEU2$ | $METEOR$ | $ROUGE_L$ |
|---|---|---|---|---|---|
| Coref Resolved SNQD | BART | 0.511272 | 0.391012 | 0.261638 | 0.498220 |
| | T5 | 0.464178 | 0.342072 | 0.227008 | 0.452510 |
| Unique CRSNQD | BART | 0.456743 | 0.337215 | 0.224961 | 0.449741 |
| | T5 | 0.450142 | 0.327546 | 0.217379 | 0.438352 |
| Moderate CRSNQD | BART | 0.499597 | 0.378458 | 0.252593 | 0.487689 |
| | T5 | 0.465117 | 0.343105 | 0.226187 | 0.453794 |

## 4.7 Human Evaluation

In human evaluation, the results are evaluated by humans. This approach has both its pros and cons. One of its pros is that the ultimate task here is to generate natural-sounding questions (like the questions that a human would ask). The best way to evaluate it would be to use different people in the evaluation process. By doing the human evaluation, we can get a precise evaluation. There is a lot of flexibility around it which we do not have in Automatic Evaluation since we can ask any questions from the people evaluating the generated data. One of its cons is that it takes lots of time and resources, which is limited in many cases [14].

Although BART's performance scores were higher than T5, in our opinion, T5 did better at generating questions. But how can that be the case:

(1) The question generated is correct but different from the targeted question.
(2) The text in questions contain words that are not in the sentence, and the model might have learned in from a possible duplicate in training dataset.
(3) The question and short answer might have little to no overlap.

As we can see in the below example, the targeted question has the word "baseball", but there is no mention of "baseball" in the sentence. The BART's generated question has the word baseball, which is correct, but it should not be like this. The T5's generated question is better than the BART's generated question as based on the provided sentence, T5 did better, but BART did better if we take the question generation as a whole.

**Sentence** THE 2017 WORLD SERIES BEGAN OCTOBER 24 AND GAME 7 WAS PLAYED ON NOVEMBER 1 , IN WHICH

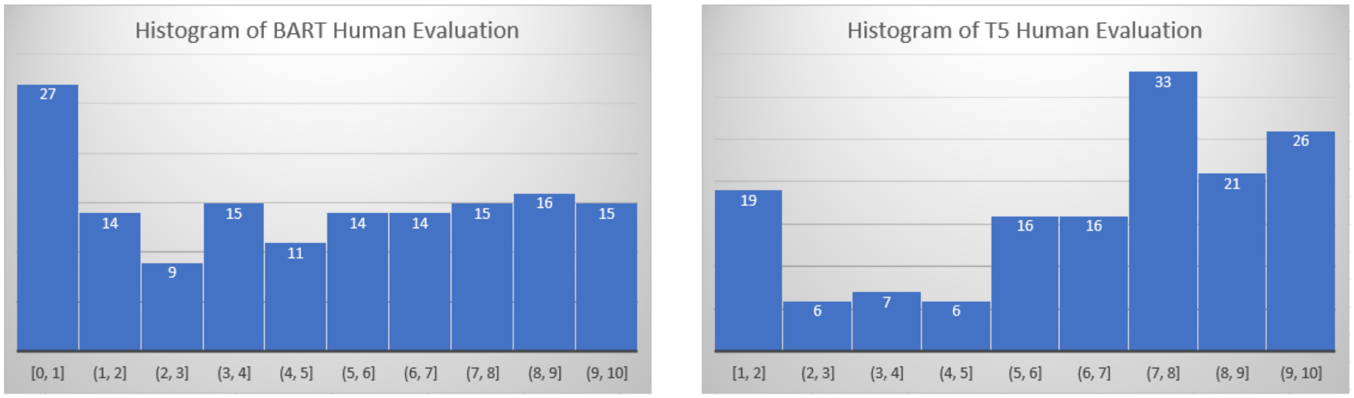

**Figure 2: Human evaluation charts**

THE HOUSTON ASTROS DEFEATED THE LOS ANGELES DODGERS , 5 - 1 , TO CAPTURE THE HOUSTON ASTROS'S FIRST WORLD SERIES CHAMPIONSHIP IN FRANCHISE HISTORY

**BART generated question** WHEN DOES THE WORLD SERIES FOR BASEBALL START

**T5 generated question** WHEN DOES THE WORLD SERIES START IN 2017

**Targeted question** HOW MANY GAMES IN THE MAJOR LEAGUE BASEBALL PLAYOFF SERIES

As we saw, the performance scores by themselves might not represent the actual performance of the models. A previous study by S. Sharma states that the automatic evaluation scores such as *BLEU* do not correlate strongly with the human judgment [12]. Natural Language Processing is about teaching the computer to 'think' like a human, so humans can be good judges in evaluating the generated questions.

We did a small human evaluation to see if the results would be consistent with the automatic evaluation results or would we get different results. Five different people did this human evaluation, each having 30 sentences, BART generated questions, and T5 generated questions, and all of these remained the same for all people. We asked them to score each generated question from 1 to 10. We selected these data randomly from the Coref Resolved SNQD.

As we can see in the Figure 2, the two histograms represent the results of the human evaluation done. As we can see, in the human evaluation, the T5 has much better results compared to the BART. T5 has 26, and BART has 15 perfect scores. In BART, most of the scores are below 6, but in T5, only 25% of the scores are below 6. The most repeated score for BART is '1' with 27 occurrences, whereas for T5 is '8' with 33 occurrences.

## 5 CONCLUSION

Summary of my contributions in this project:

(1) Developed a new Question Answer dataset named Coref Resolved SNQD for natural question generation task which can be shared with the NLP community for training neural QG models.
(2) Evaluated two state-of-the-art models using both human evaluation and automatic evaluation with BLEU, ROUGEL, and METEOR.
(3) My findings are that BART performs better in automatic evaluation and T5 performs better in human evaluation.
(4) The findings indicate more research needs to be done to improve the automatic evaluation.

This project was our first NLP or machine learning project, which made us research every bit of this project and get a solid understanding of the topic. We used Google Cloud's servers to do this project, and it was also our first time using cloud technology which helped us understand how the clouds work. By doing this project, we learned:

(1) Text processing techniques such as parsing and coreference resolution.
(2) The process of working with state-of-the-art text generation models (BART and T5).
(3) How to use BART and T5 to train models for question generation and how to evaluate them using standard measures ($BLEU$, $ROUGE_L$, $METEOR$).

There are many possible ways to expand this project, here are some of the works that we liked to do ourselves but due to the time constraint we were not able to do:

(1) Using the generated dataset on other question generation models, especially the sequence-level QG systems.
(2) Fine tuning the models used to generate questions based on the dataset given (like changing values for learning rate, number of train epochs, etc.).
(3) Checking for the possible structural grammar issues in the generated questions. This could be used for further evaluation on the models used to generate questions.
(4) Checking whether the keywords in the sentences are present in the question generated or not. This could help the evaluation process and alert the user regarding those sentences.

## REFERENCES

[1] Tina Baghaee. 2017. Automatic Neural Question Generation using Community-based Question Answering Systems.

[2] Michael Denkowski and Alon Lavie. 2014. Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Baltimore, Maryland, USA, 376–380. https://doi.org/10.3115/v1/W14-3348

[3] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. [n.d.]. *AllenNLP: A Deep Semantic Natural Language Processing Platform.* https://doi.org/10.18653/v1/W18-2501

[4] Google. [n.d.]. *Natural Question Generation Dataset.* https://ai.google.com/research/NaturalQuestions/dataset

[5] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics* 7 (08 2019), 453–466. https://doi.org/10.1162/tacl_a_00276 arXiv:https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl$_{a0}$0276/1923288/$tacl_{a0}0276.pdf$

[6] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *CoRR* abs/1910.13461 (2019). arXiv:1910.13461 http://arxiv.org/abs/1910.13461

[7] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*. Association for Computational Linguistics, Barcelona, Spain, 74–81. https://aclanthology.org/W04-1013

[8] Marta Maślankowska. [n.d.]. *How to make an effective coreference resolution model.* https://towardsdatascience.com/how-to-make-an-effective-coreference-resolution-model-55875d2b5f19

[9] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, 311–318. https://doi.org/10.3115/1073083.1073135

[10] Suraj Patil. [n.d.]. *Question Generation using transformers.* https://github.com/patil-suraj/question_generation

[11] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *CoRR* abs/1910.10683 (2019). arXiv:1910.10683 http://arxiv.org/abs/1910.10683

[12] Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. 2017. Relevance of Unsupervised Metrics in Task-Oriented Dialogue for Evaluating Natural Language Generation. *CoRR* abs/1706.09799 (2017). http://arxiv.org/abs/1706.09799

[13] Tavish. [n.d.]. *A Must-Read Introduction to Sequence Modelling (with use cases).* https://www.analyticsvidhya.com/blog/2018/04/sequence-modelling-an-introduction-with-practical-use-cases/

[14] Chris van der Lee, Albert Gatt, Emiel van Miltenburg, and Emiel Krahmer. 2021. Human evaluation of automatically generated text: Current trends and best practice guidelines. *Computer Speech Language* 67 (2021), 101151. https://doi.org/10.1016/j.csl.2020.101151

# Appendices

## A HOW TO USE THE SOFTWARE

First, we need to ensure that our system has all the necessary packages. Some of the packages that your system might not have are allennlp, spacy, and language_tool_python. CUDA is also a must for running this project which can be checked if your system currently has or not by the "nvidia-smi" command.

After downloading the zip file, which contains all the files necessary, unzip it. Then, run dataset_generation.py, which generates SNQD and CRSNQD datasets from the original dataset. Also, it generates the train, dev, and test sets required for our testings from the CRSNQD dataset with the ratio of 8:1:1 inside the directory ./aa2/data/myData.

After generating the dataset, we have to make it readable for the system. The single.py, inside the directory of ./aa2/data, does this. Running single.py changes the format of our dataset from ".txt" to ".csv", and the new formatted dataset will be saved on ./aa2/data/single.

After changing the format of the dataset, we need to tokenize the data we have for our sequence-to-sequence model to be able to train on it. For this reason, we have to run prepare_data.py in directory ./aa2. This file has three parameters to be passed on to it, which are model_type, max_source_length, and max_target_length. The only parameter that we changed during our testings was the model_type parameter between "T5" and "BART". Its default value is set to "BART". It also creates a directory of "bart_tokenizer" or "t5_tokenizer" depending on the model_type chosen, which contains the tokens. The prepare_data.py also changes the format of the dataset from ".csv" to ".pt" to make it ready for our models.

When the tokenizing is done, we run "run_bart.py" or "run_t5.py" as we have everything ready to train our models. These two files are in the directory of ./aa2. These files contain hyperparameters that we can use to fine-tune the model. It also creates a directory of "bart_model" or "t5_model", depending on which we executed. The generated directory contains the trained model, which we will use to generate our hypothesis file.

We can generate the hypothesis file by running eval.py. This file has many parameters, which are:

**model_name_or_path** For this parameter we mention the path of our model's directory which would be "bart_model" or "t5_model".

**tokenizer_name_or_path** For this parameter we mention the path of our tokens' directory which would be "bart_tokenizer" or "t5_tokenizer"

**test_file_path** We will use one of the generated files of "prepare_data.py" which is going to be either "./data/test_data_bart.pt" or "./data/test_data_t5.pt".

**model_type** Which would be "bart" or "t5"

**output_path** For this parameter we mention our desired path and name of the hypothesis file like "./hypothesis_t5.txt"

Inside the hypothesis file would be the system's generated questions which we now have to evaluate them using automatic evaluation process. To do this automatic evaluation, we used "nlg-eval" and it also has two parameters that we have to give to the system, which are as follows:

**hypothesis** The path of the hypothesis file that we got from running eval.py.

**references** The path of the test dataset which should be inside directory ./aa2/data/myData.

After running "nlg-eval" with the right parameters, the metrics of the dataset should be printed in the console.