Improving Relevance of QA Responses for Query Inputs

Thivakkar Mahendran tmahendran
Vincent Pun cpun
Kathryn Ricci kdricci
Apurva Bhandari apurvabhanda
Somin Wadhwa sominwadhwa

Abstract

BERT can be fine-tuned to build question answering systems and systems for various other NLP tasks. However, BERT is unable to provide answers to queries search engine users typically input, which are usually brief or fragmented sentences. To address this problem, we propose a two pronged approach: 1) fine-tuning the BERT QA model on queries 2) developing an NMT model to convert queries to questions, which are then used as input to the standard BERT QA model. Our code is publically accessible at https://github.com/sominwadhwa/cs585-q2q.

1 Introduction

Modern search engines (like Google, Bing, DuckDuckGo) are getting increasingly better at interpreting queries and extracting relevant text from a huge corpora of results. For example, a Google search for "Make Tea" results in an instruction snippet on how we should go about making tea. But a query like "nearest RMV" results in no extracted text; rather it gives us list of web pages with the relevant information. However, searching for "Where is the nearest RMV?" gives us the full address of the nearest RMV office. Recent advances in the usage of state-of-the-art natural language models like BERT (Devlin et al., 2018) to extract contextual information from vast portions of text have made such IR tasks possible (Singh, 2018). On QA-type tasks with reading comprehensions for instance, we already seem to surpass human level performance (Rajpurkar et al., 2016). However, if we frame the questions as natural language queries, something what an actual user is more likely to input, the QA systems do not seem to perform well on queries. To address this problem, we propose two approaches: 1) fine-tuning the BERT QA model on queries 2) developing a NMT model to convert queries to questions, which are then used as input to the standard BERT QA model.

Apart from the obvious application in search engines, a Q2Q system such as that used in our NMT approach has applications in chatbots (question-answering) and communityQA (Zhao et al., 2011).

Key Contributions: As a starting point, we take an exisiting model that is known to work well on QA type tasks (BERT) and a QA dataset (SQuAD), generate queries using the questions available and assess performance of these queries on the QA model by evaluating responses. Surprisingly, we observe a huge dip in performance. Then we go on to fine tune the QA model using our queries, and the performance improves a little, but not as much as we thought. So we go on to train a Neural MT model with attention to automatically generate questions given the queries. While it isn't as good as we had expected, but it still gives reasonable results (see attention maps in Error Analysis below). On top of that, we also conducted some probe tasks by modifying our queries in different ways to investigate what our model actually learns.

2 What you proposed vs. what you accomplished

Proposed & **accomplished** ☺

- Collect and preprocess QA datasets - SQuAD, HotpotQA, Google's Natural Queries dataset
- Generate meaningful queries from questions + assess the right dataset (SQuAD) through data exploration

- Evaluate performance of queries based on SOTA QA models + fine tune them to work better on queries itself.
- Train a neural machine translation model to generate questions from queries.
- In depth error analysis on the query generation process and the outputs of our model Q2Q model.

Proposed but couldn't accomplish ☹

- Q2Q model output is reasonable but not as good as it should be. Sometimes there is a lot of redundancy in the generated output (see Error Analysis below).
- We also wanted to generate queries by a rule based approach to see how that fares against an automated appraoch adopted by us, but it was extremely time consuming and we believed that we could better invest our time in analysing our results.
- We also wanted to try a more complex approach of generating questions by taking < $Query$+$Context$ > as input to the MT model & then generate questions using some form of constraint decoding. We simply ran out of time. Our baseline models + analysis of results proved to be much harder and more time consuming that we initially estimated.

Didn't propose but did it anyway ☻

- Per feedback from the poster session, we designed certain probe tasks by modifying our queries by adding/removing certain key question words to see how they affect the overall performance, and the results were quite surprising.

3 Related work

Information extraction has been an active area of research in NLP (Tan et al., 2019; Xiong and Sun, 2018; Zhong et al., 2017) in the recent past. Domain specific question answering tasks, like CommunityQA (Wu et al., 2018; Nakov et al., 2017; White et al., 2015) have predominantly served as a defacto standard for evaluating how good modern lanuage models are at extracting relevant information from large corpora of text. Towards this end, SQuAD (Rajpurkar et al., 2016) and HotpotQA (Yang et al., 2018) have been curated and evaluated on human-level performance to serve as benchmarks against which most modern language models are evaluated. And within the last few years, it appears that we've achieved super-human performance on these QA tasks using natural language models that rely on contextual represenations of input (Peters et al., 2018; Lan et al., 2019; Zhang et al., 2019b,a). Query to question conversion to improve information extraction, however, was first suggested by (Lin, 2008). Lin directly linked this idea to improve query expansion in CQA communities. This idea was further amended by (Zhao et al., 2011), where they follow a template based approach by generating <*query, question*> pairs from CQA search logs. But this framework, again, relies on the availability of a massive corpus of questions. (Kalady et al., 2010), although with a different overall objective, proposes a method to create questions from well formed sentence parse-trees and named entity recognition (NER) models. (Kumar et al., 2018) is the first recent effort in the direction of incorporating machine translation models into the context of converting queries to questions. In their approach, they try to convert web-queries to well formed questions using Statistical Machine Translation (Cho et al., 2014) and Neural Machine Translation Models (Sutskever et al., 2014; Bahdanau et al., 2014). However, even their approach is limited by the evaluation methodology employed. Their primary method of evaluation is the use of BLEU score, which has its own set of limitations (Post, 2018). While they also report human evaluation on certain parameters like grammatical correctness, however, those parameters are often irrelevant in this context since state of the art language models are perfectly capable of extracting relevant responses even the questions being fed are do not hold perfect grammatical correctness.

4 Your dataset

We started our data exploration by looking at three different QA datasets -

1. SQuAD - Stanford Question Answering Dataset (Rajpurkar et al., 2016)
2. HotpotQA (Yang et al., 2018)
3. Google's Natural Questions dataset (Kwiatkowski et al., 2019)

Google's natural questions dataset was the most straightforward in terms of obtaining natural language queries problem is that we did not have associated passage with each query. Hence we cannot feed this into BERT's question answering system without crawling the web for related context. Also, Google Natural Questions are actual web queries and they are mostly fully-formed questions, unlike what we expect of a typical human generated query (Table 1).

HotpotQA and SQuAD both looked promising but upon further inspection, we noticed that queries generated using our approach on HotpotQA were, on average, longer in length than a standard natural language query that a human might input. Another issue with our dataset has to do with some of the properties of the SQuAD dataset, which we ultimately chose, out of the three possibilities, to generate our query dataset from. Many questions in SQuAD are un-query-like due to depending heavily on context (e.g., Example S4 in Table 1) or being more complex/verbose than a natural query. Although this is more of an issue with HotPotQA questions, a typical example of which is shown in H2 of Table 1, it is also observable in the average query and question lengths shown in Table 3. In these respects Google's Natural Questions dataset would be closer to ideal for our application; however, at the time that we were deciding which dataset to use, there were two reasons why we did not use the Natural Questions dataset:

- They perform certain heuristics and the goal of these is to discard a large proportion of queries that are non-questions, while retaining the majority of queries of 8 words or more in length that are questions. This would still not resemble queries that we feed into search engines.
- The corresponding Wikipedia web-pages would have to be crawled and processed as in the NQ dataset, only the links to the Wikipedia pages were available.

We have now described the models that we have used in our work. The train-set of the model has input in the format [input query, passage, answer]. We have trained the model with three kinds of input queries - questions, generated queries, generated queries with prepended Wh-words(referred to as "Query with wh-words"). The dev-set and test-set have input in the format [input query, passage] and the output is the answer. The table 2 is a description of what we refer to the particular model as, where a model is defined as ["input query" in training set,"input query" in training set].

We decided to use the database of query and question pairs we generated from SQuAD. At the end of our preprocessing, we ended up with 130319 training (+ validation) and 11873 test examples.

4.1 Data preprocessing

One aspect of our project that proved to be extremely challenging was that, due to the difficulty of obtaining web query logs from which we could collect actual queries and questions, we needed to create our own synthetic queries. To investigate some options, we took questions from the three question answering datasets and aimed to obtain stripped-down versions of the original questions that resembled what a human might type into a search engine. The approach we took involved stop word removal and lemmatization. Some examples of the results using these different datasets can be seen in Table 1. In reality, however, a more complex transformation happens between question and natural query: for example, vocabulary can change, word order can be shifted, and words like "to" (which we removed) can be included in the query. (Kumar et al., 2018) provide some examples of queries generated from actual web-logs along with their associated questions which, at times, differ considerably from the ones we have generated.

5 Baselines

We created two baselines to use for both the fine-tuned model and the NMT model. Table 4 shows the performance of the baselines. This section discusses how the baselines are created, and how they are used. All fine-tuning is done using code provided by Huggingface Transformers (Wolf et al., 2019).

The first baseline (Question-on-Question) is an uncased $\text{BERT}_{\text{LARGE}}$ model fine-tuned on the original SQuAD 2.0 question dataset (Rajpurkar et al., 2016), and evaluated with the original dev set. The second baseline (Query-on-Question) uses the same model, but is evaluated with our

	SQuAD
S1	A. Who did Beyonce tie with for the most nominations in a year? B. beyonce tie nomination year
S2	A. When did Winstron Churchill form his government? B. winstron churchill form government
S3	A. What part of the airway is especially effected by neutrophils? B. part airway especially effected neutrophil
S4	A. What does Cantonese have more of among Chinese varieties? B. cantonese among chinese variety
	Google Natural Questions
G1	A. who has the most followers on instagram in the world B. follower instagram world
G2	A. what causes a dead zone in the ocean B. cause dead zone ocean
G3	A. where is the original statue of david in italy B. original statue david italy
	HotpotQA
H1	A. Which state does the drug stores, of which the CEO is Warren Bryant, are located? B. state drug stores, ceo warren bryant, located
H2	A. What American country music singer-songwriter, born in May of 1942, sang a duet with her ex-husband the same year that he released the song "The Battle?" B. american country music singer-songwriter, born may 1942, sang duet ex-husband year released song "the battle"
H3	A. Who invented the type of script used in autographs? B. invented type script used autograph

Table 1: Examples of Queries Generated on SQuAD, HotpotQA, Google's Natural Questions Dataset.

own dev query set.

The F1 and EM scores dropped significantly for the Query-on-Question baseline. This is expected because many words are omitted in the queries. The NMT model generates questions from the queries, which we then evaluate on the model fine-tuned on SQuAD 2.0 question dataset. This should output scores that are better than the Query-on-Question baseline. If the generated questions are close to the original questions, the scores should be close to the scores for our model. We discuss the results in Section 7.

6 Your approach

To address this problem, we created a model by fine-tuning an uncased $\text{BERT}_{\text{LARGE}}$ model with our query dataset and evaluating with the test query dataset (Query-on-Query). The same configuration was used, fine-tuning is performed for 2 epochs with a batch size of 3 (based on multiple encounters with memory issues). The learning rate is set to 3e-5. Devlin et al. use a batch size of 48 and a learning rate of 5e-5. We reduced the batch size due to memory constraints. We use the same train/dev split from SQuAD 2.0, with 130,319 questions in the train-val (80-20) split, and 11,873 questions in the final test split.

We also considered this a machine translation problem, where queries are in the source language and generated natural languages are in the target language. We also built a model that follows a very similar approach, using our own dataset. We first created a neural machine translation (NMT) model using the Keras library (Chollet, 2015) which was trained on the SQuAD dataset. The model's architecture was LSTM with RepeatVector (Figure 1).

Since the output wasn't very good, we modified our NMT model to include sequence-to-sequence encoder-decoder attention architecture with 512 hidden units. We used the Pytorch library for our new model. While we wanted to, but couldn't implement complex models like bi-directional

Dev set format	Format fine-tuned on	Referred to as in this work
Question	Question	Question-on-Question
Question	Question	Query-on-Question
Query (our approach)	Query	Query-on-Query
Query with wh-words	Query with wh-words	Query-Wh-on-Query-Wh
Query with wh-words	Question	Query-Wh-on-Question
Query with wh-words	Query with wh-words	Query-Wh-on-Query
Wh-words only	Question	Wh-on-Question
Wh-words only	Query	Wh-on-Query

Table 2: Col 1: Description of what queries we gave as validation set input; Col 2: Description of what the training-set to the model was; Col 3: The term we use to refer to this model. (An important terminology to refer to in future!)

Average Length (# tokens)			
	Questions	**Queries**	**Passage**
SQuAD	9.9	5.3	117
HotpotQA	17.6	10.4	887

Table 3: Average lengths of questions/queries in terms of the number tokens

	F1	**EM**
Question-on-Question	82.4058	79.2891
Query-on-Question	62.1541	59.4795

Table 4: F1 and EM scores for the Question-on-Question and Query-on-Question baselines. There is a significant but expected drop in the F1 and EM scores when the dev query set is used, compared to Question-on-Question.

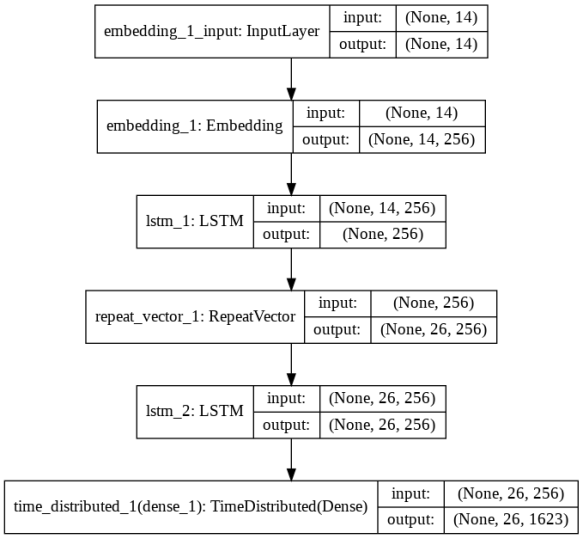

Figure 1: Baseline NMT without Attn

LSTMs or a transformers-based model because these models take a long time and computational-resources to run and experiment. We chose sequence-to-sequence encoder-decoder attention model because it is doable and faster to prototype and evaluate. One thing we noticed is that the NMT model is aptly able to put back a wh-word back into the question, where the input is a query (remember when we generated the query, the stop word removal process removed the wh-word!). We expect that, using these generated questions, the QA model performs better than the baseline BERT QA system. Depending on the quality of the translations, the scores should be close to the scores for the Question-on-Question baseline because we want the queries to be answered with just as much accuracy.

The primary libraries that we used were Huggingface Transformers (Wolf et al., 2019), PyTorch (Paszke et al., 2019), Keras, and NLTK. Keras and PyTorch were used to build the neural machine translation (NMT) model. The NLTK library (https://github.com/nltk/nltk) was used synthetic dataset generation. The Huggingface Transformers (https://github.com/huggingface) library, which is a library on top of PyTorch that lets you build networks with transformers, helped us with the fine-tuning of BERT. We primarily used Google Colaboratory for the project. We chose Google Colab because it is free to use and has an

option of multiple people working on the project at the same time. For the project, training the model required a high-performing computer and Google Colab has dedicated virtual computers that have high specifications and GPU and TPU power which was useful for the coding and compiling purposes. We had to use Google Cloud Platform (GCP) for fine-tuning and evaluating the baseline model because we reached the RAM limit of 25GB in Google Colab. Running the code on GCP was faster but we used GCP for only the essential parts of the project because GCP is not free and it would have been very expensive to run our whole project on it.

The challenge that we faced was the absence of a dataset. We did manage to generate a synthetic dataset, but it is still not an exact represenation of natural queries. For instance, among the NLTK stop-words, there is the word "most", which can be a critical word in a query but is removed by stop-word removal.

7 Error analysis

7.1 Analysis of BERT Performance on Query Datasets

We examined approximately 100 examples from several combinations of models and development set formats. The models included the BERT question-answering model fine-tuned on questions, queries, queries including question words, and the question words alone (generated as described in the Data section). Table 5 shows the F1 and Exact Match scores for all of the evaluation methods.

Our first observation was that our query models have even more difficulty than the original baseline question model predicting whether a question/query is unanswerable, as shown in Table 5. Most of the overall exact match (EM) errors that the models make are in this category, from 64.2% for Question-on-Question to 78.9% for Query-on-Question. In particular, Table 5 shows that Query-on-Question appears significantly biased towards predicting unanswerable as compared to the other models. This could be because many of the words that it depends on to answer questions have been removed from the queries, so it doesn't recognize the queries as answerable. Meanwhile, apart from the Question-on-Question baseline, the Query-on-Question model used in our approach has the best results, though still significantly worse than the Question-on-Question baseline.

The errors made by the Question-on-Question baseline model include a few noticeable types:

- Errors where the model seems to ignore crucial words in the question, while associating the Wh-word with another word in the question that is close-by (see example 1).
- Errors due to faulty coreference resolution (either missing or erroneous - see example 2)

Example 1:
Question: Who ruled the country of Normandy?
Correct answer: <unanswerable>
Wrong answer (Question-on-Question): Richard I of Normandy.
Context: "The Duchy of Normandy, which they formed by treaty with the French crown, was a great fief of medieval France, and under Richard I of Normandy was forged into a cohesive and formidable principality in feudal tenure."

In this example, the model seems to ignore the word "country" in the question in order to associate a "who" (Richard I) with "Normandy", which is close-by in the sentence.

Example 2:
Question: How many Huguenots were killed in Toulouse?
Correct answer: "Nearly 3,000"
Wrong answer (Question-on-Question): <unanswerable>
Context: "...Nearly 3,000 Protestants were slaughtered in Toulouse alone..."

The full context refers to Protestants and Huguenots interchangeably, but the model fails to recognize the coreference, so it fails. It is possible that Question-on-Question simply did not have enough data to master these types of examples.

Similar to Query-on-Question, there is a drop in the F1 and EM scores for the Query-on-Query evaluation, although to a lesser extent. In examining the kinds of questions Query-on-Question and Query-on-Query both fail at, a few common characteristics arise. First, most belong to the category of decisions involving unanswerable questions described above. Second is the appearance of a preposition in the original question which was

Model	True Neg.	True Pos.	False Neg. (FN)	False Pos. (FP)	Acc.	Total fails (EM)	FP & FN / Total Fails
Question-on-Question	4834	5336	592	1111	0.8566	2653	0.6419
Query-on-Question	5040	2958	2970	905	0.6736	4910	0.7892
Query-on-Query	4152	5078	850	1793	0.7774	3712	0.7120
Query-Wh-on-Query-Wh	3885	4920	1008	2060	0.7416	4246	0.7226
Query-Wh-on -Question	4546	4327	1601	1399	0.7473	3956	0.7583

Table 5: Errors concerning unanswerable questions: errors made by marking an answerable question unanswerable (false negative) or answering an unanswerable question (false positive). "Total fails" is the total number of exact match fails made by the model, including errors made by giving the wrong answer to an answerable question.

removed in the creation of the query. And for all, the queries do not include the question word -

Example:
Question: When were the Normans in Normandy?
Query: norman normandy
Correct answer: ['10th and 11th centuries', 'in the 10th and 11th centuries']
Wrong answer (Query-on-Query): a region in France
Wrong answer (Query-on-Question): Normans

The above query example is missing 'when' (a question word) and 'in' (a preposition). It is clear that the models are answering a question corresponding to a different question word than "when", because the query is ambiguous. To see if the missing question word was the main problem for these models, we created the Query-Wh dataset and ran it on several models (see Table 5).

One puzzle we faced was the low performance of Query-Wh-on-Query-Wh, although it did answer the above example correctly. Consider following example, which Query-Wh-on-Query-Wh gets wrong but Query-on-Query and Query-Wh-on-Question get right:

Example:
Question: What naval base fell to the Normans?
Wh-query: what naval base fell norman
Correct answer: <unanswerable>
Wrong answer: Dyrrachium
Context: "Some time later, Dyrrachium—one of the most important naval bases of the Adriatic—fell again to Byzantine hands."

We hypothesize that the model fine-tuned on questions learns the relationship between the question word and words like prepositions that are excluded from queries. For example, it understands that there is a sense in which something (a "what") might "fall to" as well as simply "fall". It still retains this knowledge in Query-Wh-on-Question, improving performance over Query-Wh-on-Query-Wh. Meanwhile, the model fine-tuned on wh-queries seems to have learned to in many cases simply return the nearest words that are a "what" and defer to that reasoning (which may work well much of the time) instead of the reasoning used more successfully on the dev set by the model fine-tuned on the original queries. If there is no "what" word nearby, it seems to default to unanswerable (it doesn't succeed on examples with long-term dependencies). But the method seems to work in the case in the "When were the Normans in Normandy?" example that Query-Wh-on-Query-Wh answered correctly.

The performance of Query-Wh-on-Query is approximately the same as that of Query-on-Query because the Query model learned no weights for the question words, so the addition of the question words has little effect. Therefore, we conclude that the missing question words are not the sole or primary reason for the performance gap between the Query-on-Query model (our approach) and the baseline Question-on-Question model. Instead, it is most likely the absence of words like prepositions that we removed during query creation, with the absence of the wh-words perhaps contributing.

In addition, to see the effects on performance when only the question words exist, we performed an evaluation with a dev dataset that contains only the question words, and used these

queries on the Question and Query models. The Wh-on-Question evaluation achieves an F1 score of 34.44. With much of the context removed from the questions, the model is unable to infer what the questions are and hence cannot predict answers in most cases. It considers most questions unanswerable, making the F1 and Exact Match scores for unanswerable questions high. In reality, the model predicts very poorly on questions that it considers answerable, save for a few cases where the question words alone can pinpoint the contexts in the passages. A similar effect is observed in the Wh-on-Query evaluation, which achieves an F1 score of 42.37. The model has no knowledge of the question words and hence considers even more questions unanswerable, increasing the F1 score as a result. It can be seen from these examples, then, question words are an important indicator of what makes a question answerable, corroborating the hypotheses above. An example correct response from Wh-on-Query where the context is obvious -

Example:
Question ID: 5737a84dc3c5551400e51f5a
Question: Why are some forces due to that are impossible to model?
Wh-Query: why
Correct answers: ['gradient of potentials', 'macrophysical considerations that yield forces as arising from a macroscopic statistical average of microstates', 'gradient of potentials.', 'gradient of potentials']
Answer from Wh-on-Question: macrophysical considerations that yield forces as arising from a macroscopic statistical average of microstates
Context: "This is often due to macrophysical considerations that yield forces as arising from macroscopic statistical average of microstates."

7.2 Analysis of NMT Model

We examined the results from the NMT model. Using the dev query set, we generated 11,788 question. 85 questions could not be generated. The NMT model is able to construct questions from the queries for the training set, although sometimes with peculiarities. Figure 2 shows the loss curve of the model during training. Table 6 shows samples of generated questions from their queries in the training set. We observe that the model is able to learn that the translation target is a question and inserts question words, and in most cases, ends the questions with a question mark. The model however is often unable to complete the questions or often repeats words.

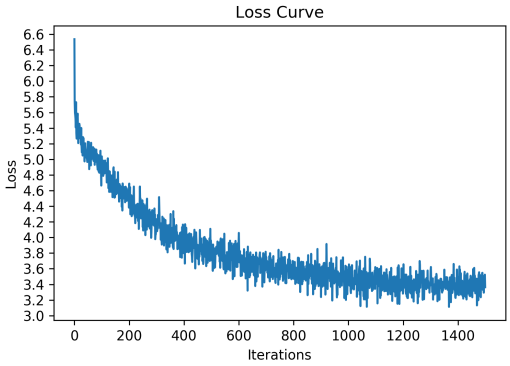

Figure 2: Loss Curve for NMT model with Attention

The samples suggest that the model learned which question word is to be used for a question, based on the cues in the query. For example, a query that starts with the word "year" is translated into a question beginning with "in what year," (samples 1, 2, 3), a query about an object (in the concrete or abstract sense) is translated into a question with the question word "what" (sample 4), and a query about people is translated into a question with the question word "who" (sample 5, 6).

Figure 3 shows the attention map of sample 4. It indicates that "new" is correlated with an object and hence there is a darker block in the "what" row. Figure 4 shows the attention map of sample 6. It indicates that the model has learned a correlation between the word "director" and the word "movie."

There are cases where the attention maps do not reveal any particular information. For example, Figure 5 and Figure 5 and show attention maps where the weights are heavily skewed towards the end of the queries.

We hypothesize that the model repeats words and fails to complete the questions because of a choice of decoding strategy that we used. We used a beam search strategy, which would go down a wrong path if the combined probability of a path is higher than that of another path, causing repetitions.

However, when the same model is used to generate questions from the dev query set, it breaks down. Table 7 shows samples of generated questions from their queries on the training set. We ob-

Sample	Query	Predicted Question
1	year american revolution .	in what year did the revolution revolution revolution?.
2	year republic ireland formed .	in what year did the republic republic formed the republic?.
3	year america join world war .	in what year did the world war join the world war?.
4	political machine controlled new york politics era .	what era of the the new the new new new??...?
5	first roman babtized .	who was the first roman?.
6	director of the movie .	who was the director of the director of the.

Table 6: Samples of generated questions from queries on the training split. The model is able to insert question words in all cases, but often cannot complete sentences properly.

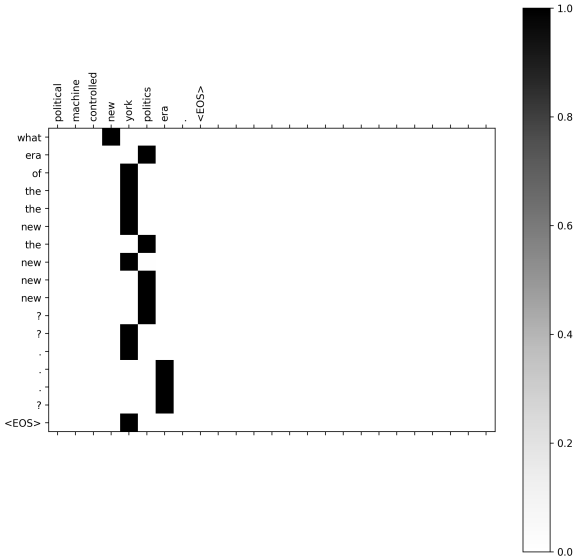

Figure 3: Spurious Output from NMT Model

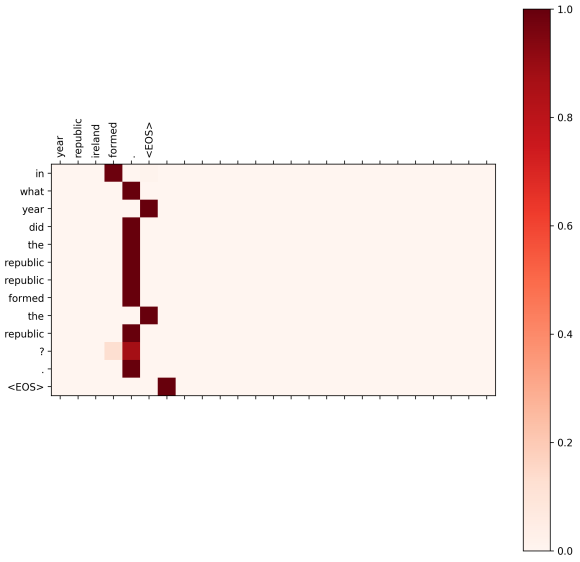

Figure 4: Question Output from NMT Model

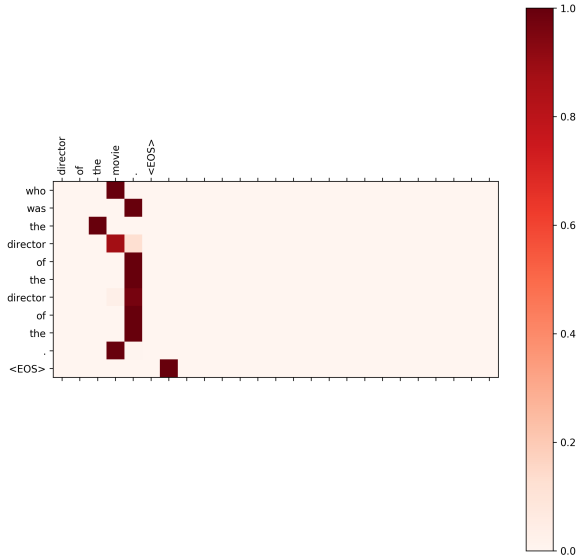

Figure 5: Question Output from NMT Model

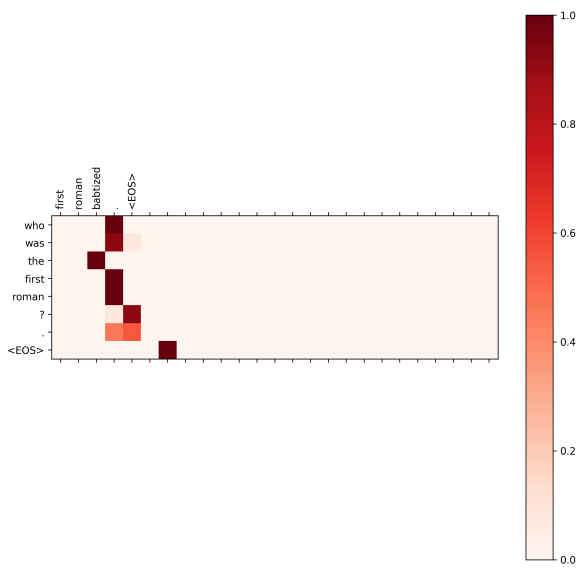

Figure 6: Question Output from NMT Model

serve that many of these generated questions use words from the training set. This is likely because we did not have enough words in the vocabulary for the model to learn during training, and hence the model generates nonsense.

7.3 Performance on Model Fine-Tuned with Questions

We fed these questions into the BERT model fine-tuned on the question dataset for evaluation. Table

Sample	Query	Predicted Question
7	principality william conquerer found .	in which region language language located located assimilted ? .
8	island coast asia .	when what bought located bought bought normans ? .
9	crime rate show correlated society .	when th region region located unless th assimilted located ? ? ?
10	known working portfolio capture theory .	on which region region located located located located located ? ? ? ? ?
11	obediant ispah rebellion .	on measures leader techniques located normans . ?
12	simple unicellular organism lack .	when located italy smaller integer nephew located nephew located ? .

Table 7: Samples of generated questions from queries on the dev split.

F1 Score	49.0300
EM Score	48.9650
HasAns_F1	0.4191
HasAns_EM	0.2889
NoAns_F1	97.4763
NoAns_EM	97.4763

Table 8: Scores from running BERT fine-tuned on the dev query set with questions generated from the queries using the baseline NMT model.

8 shows the results.

For the questions that the model considers unanswerable, it performs poorly. Consider the examples below, the model predicts the answers correctly despite the undecipherable questions. The passages associated with those questions are rather short, and we suspect that the question words or the words in the generated questions gave enough clues for the model to predict the correct answer.

Example:
id: 56de1563cffd8e1900b4b5c3
Question: What was the naval base called?
Generated question: denotes what region attacked attacked located normans .
Correct answers: ['Dyrrachium', 'Dyrrachium', 'Dyrrachium']
Answer from model: Dyrrachium
Context: "The further decline of Byzantine state-of-affairs paved the road to a third attack in 1185, when a large Norman army invaded Dyrrachium, owing to the betrayal of high Byzantine officials."

Example:
id: 56de4a474396321400ee2786
Question: Where are Jersey and Guernsey
Generated question: denotes located which located located located . ? .
Correct answers: ['Channel Islands', 'the Channel Islands', 'the Channel Islands']
Answer from model: Channel Islands
Context: "The customary law of Normandy was developed between the 10th and 13th centuries and survives today through the legal systems of Jersey and Guernsey in the Channel Islands."

8 Contributions of group members

- Kathryn: Data collection/analysis, preprocessing (including query generation) & a lot of error analysis.
- Thivakkar: Built the baseline NMT model & data preprocesssing and considerable writing.
- Apurva: Formulation of the overall idea of the problem along with initial literature review and analysis for future work in this direction.
- Vincent: Fine tuning original QA models based on BERT, evaluation of results on all our models and a lot of error analysis.
- Somin: Built the Q2Q machine translation model with attention, visualizations, lot of writing and some error analysis.

9 Conclusions

In this work, we attempted to highlight and tackle two key issues with modern state-of-the-art question-answering models. First, we did a thorough analysis of how these models perform on sentence fragments (queries) rather than well formed questions. Second, we attempted to formulate the conversion of Query to Questions as a machine translation problem & used a Seq2Seq with attention-style architecture to do so. Our key takeaways from this project were -

- It is incredibly hard to curate good quality queries from existing QA databases. While we feel that we did a reasonable job in generating queries, there could be a better representative dataset for the task.

- When we fine-tuned BERT on the query dataset, we could improve the performance by 10.9 F1 points.
- One interesting observation that we made: When we fine-tuned BERT on queries, that had the wh-word prepended to the query and tested it on the same queries, we found the F1 score to be 68.94. This is in contrast with the score obtained on BERT fine-tuned on initial query(without the wh-word) and tested on the same: 73.06. We were expecting that when we add the wh-word to the query, the performance of the model should go up because the query then becomes more question like and therefore should be easier to answer.
- We need to try to generate a better NMT model for the query to question conversion.

Given more time and resources, one of the things we would definitely want to try is to come up with better methods to generate queries using some statistical or heuristic methods. Another thing would be to try more complex models like bidirectional RNNs or transformers to capture information from queries & context to generate more meaningful questions.

References

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation.

Chollet, F. (2015). keras. https://github.com/fchollet/keras.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding.

Kalady, S., Elikkottil, A., and Das, R. (2010). Natural language question generation using syntax and keywords. In *Proceedings of QG2010: The Third Workshop on Question Generation*, volume 2. questiongeneration. org.

Kumar, A., Dandapat, S., and Chordia, S. (2018). Translating web search queries into natural language questions. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Kelcey, M., Devlin, J., Lee, K., Toutanova, K. N., Jones, L., Chang, M.-W., Dai, A., Uszkoreit, J., Le, Q., and Petrov, S. (2019). Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations.

Lin, C.-Y. (2008). Automatic question generation from queries.

Nakov, P., Hoogeveen, D., Màrquez, L., Moschitti, A., Mubarak, H., Baldwin, T., and Verspoor, K. (2017). SemEval-2017 task 3: Community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 27–48, Vancouver, Canada. Association for Computational Linguistics.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations.

Post, M. (2018). A call for clarity in reporting bleu scores.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text.

Singh, S. (2018). Natural language processing for information extraction.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks.

Tan, F., Cascante-Bonilla, P., Guo, X., Wu, H., Feng, S., and Ordonez, V. (2019). Drill-down: Interactive retrieval of complex scenes using natural language queries.

White, R. W., Richardson, M., and Yih, W.-t. (2015). Questions vs. queries in informational search tasks. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 135–136, New York, NY, USA. ACM.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., and Brew, J. (2019). Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Wu, W., Sun, X., and Wang, H. (2018). Question condensing networks for answer selection in community question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1746–1755, Melbourne, Australia. Association for Computational Linguistics.

Xiong, H. and Sun, R. (2018). Transferable natural language interface to structured queries aided by adversarial generation.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., and Manning, C. D. (2018). Hotpotqa: A dataset for diverse, explainable multi-hop question answering.

Zhang, Z., Wu, Y., Zhao, H., Li, Z., Zhang, S., Zhou, X., and Zhou, X. (2019a). Semantics-aware bert for language understanding.

Zhang, Z., Wu, Y., Zhou, J., Duan, S., Zhao, H., and Wang, R. (2019b). Sg-net: Syntax-guided machine reading comprehension.

Zhao, S., Wang, H., Li, C., Liu, T., and Guan, Y. (2011). Automatically generating questions from queries for community-based question answering. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 929–937, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Zhong, V., Xiong, C., and Socher, R. (2017). Seq2sql: Generating structured queries from natural language using reinforcement learning.