Question Answering on Tabular Data with NLP

Jiyang Ge
jg6064@nyu.edu

Junrong Zha
jz3741@nyu.edu

Yunya Wang
yw4509@nyu.edu

Abstract

In recent years, using natural language model to solve question answering system on structured data has been a popular field. In this paper, we present two new model structures built on top on TaBERT (a pre-trained language model developed by Facebook) and train it on WiKiSQL dataset to compare their performances.

1 Introduction

Recent years, there have been a burgeoning trend of questions answering task in the NLP field, especially in the area of understanding the semi-structure tables. Traditonal way to construct a question-answering system over database is transforming the natural language to logic form such as SQL, which computer can execute. Guo and Gao [2019] mentions another approach as to inject the logic form step into the deep model. Our goal is to build a question-answering system to directly fetch answer from the tabular content without converting the questions into SQL. Being inspired by Herzig et al. [2020] who presents TAPAS, a model uses BERT-based architecture as encoder and uses two classification layers to select the cell, we follow similar structures. Mentioned by Yin et al. [2020], we uses TaBERT, a pre-trained language model that learns representations for natural language sentences and semi-structured tables together, as encoder to generate context and table information.Then make attempt to use both binary classification and multi-class classification as decoders. We also use different methods to add attentions to the multi-class classification. In addition, Zhong et al. [2017] mentions sequence to sequence model could be a method to generate structured queries, therefore we also try sequence to sequence model with self attention as the decoder. In general, classification model performs better than sequence to sequence model in terms of both accuracy and f1 score. The best fine-tuned model is multi-class classification model constructed by applying attention to column encoding and context encoding separately. As it achieves highest accuracy of 0.9344 and highest f1 score of 0.9345.

2 Related Work

To construct a question-answering system over database, semantic parsing is widely used in recent years. Semantic parsing is a task that transform the natural language to logic form which computer can execute. In question answering, semantic parsing can be used to transform from questions to SQL (NL2SQL), which can be executed in the database system to generate the answers. In Table2answer, different from previous method, they do not use the logic form. They inject the logic form step into the deep neural network model. The main motivation is that if human could generate the answers without an explicit logic form by only reading the schema in the database and find the answer, so the does the neural network. They use BERT-based model to encode questions and columns and train the model by classify the final pointer to the cell of the answer.

Earlier this year, Facebook published a new paper introducing an another way of questions and columns encoding, (Yin et al. [2020]) which is also built on top of BERT, and jointly learns contextual representations for utterances and the structured schema of database tables. The output of TaBERT is fixed dimension vector representation of the table and context (Zhong et al. [2017]) and we are incorporating this as our encoder in our project.

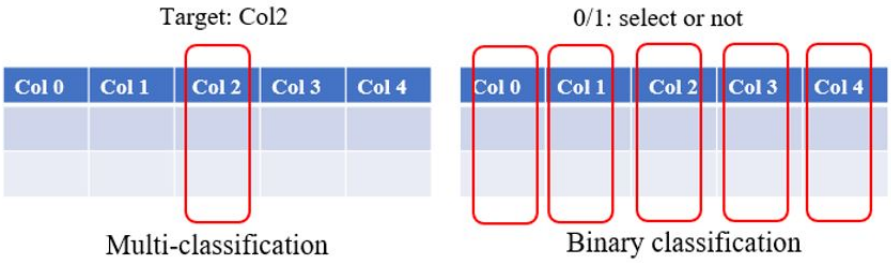

Figure 1: The difference between multiclass classification and binary classification. In the multiclass classification, we select the target column, while in the binary classification, we assign 0/1 to each column, which represents been selected or not.

Additionally earlier this year, Google released a paper on this subject as well called TaPas. They brought up an idea of row and column classification for the question and answering system for the tabular dataset with their own defined encoder. And we are incorporating this idea as one of our decoders in our project.

3 Problem Definition and Algorithm

3.1 Task

Given the dataset contains hand-annotated examples of questions and SQL queries distributed across tables, we first build a data loader to transform the information of tables and questions into table object and context object, including concatenating column type, column name, and rows into one object, tokenization of table and context, padding max sequence for answers, ect. In addition, we use these data loader to get the information in SQL queries as labels to indicate which cell the answer is. In the original dataset, the table contains questions of single cell selection and questions requiring aggregation like summation, averaging and ranking. Given the complexity of the data and the model, we only include questions of single cell selection and exclude the aggregation questions from our dataset.

As for the main part of the model, we feed our processed input into TaBERT, which is our encoder [1] to get a fixed dimension vector representation of the table and the context; after that we construct several decoders. We implement two types of decoder to tackle the problem, one is classification model and the other one is sequence to sequence model. For the classification approach, we first try to make binary classification for each column as 1 stands for the answer is in the column and 0 otherwise. Moreover, we try multi-classification as a numerical value stands as the index of the column where the answer is. Then we apply row classification to indicate the position of exact cell. For the sequence to sequence approach, we use the encoder output as the input to the Seq2Seq model and treat the answers like a sequence prediction of tokens. For example, answer like 'New York' will be tokenized into sequence of ['New' 'York'] as our target. And we also incorporate masking and self attention in it as well.

In terms of pros and cons of the two approaches, in general, both binary and multi-class classification models could be conducted more easily than sequence to sequence model due to less complexity in architecture. Also, classification model that combines column and row classification is able to determine the cell position, while it is harder to train the Seq2Seq model given the complex input and model structure. However, Seq2Seq model has the ability to have input as different sizes while classification model needs inputs with the same length. While keeping the same length for input, the cutting off for some sentences may lead to loss of information and the padding might add noise to the model.

3.2 Algorithm

3.2.1 Column Classification and Row Selection

In this section, firstly we describe the details of our TaBERT-based model to solve the column classification task. Then we describe the procedures for row selection.

[1] https://github.com/facebookresearch/TaBERT

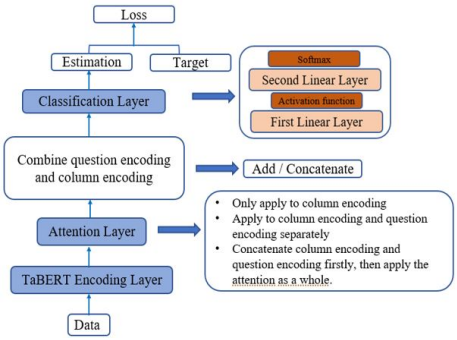

Figure 2: The detailed framework for classification task. The attention layer is only used for multiclass classification.

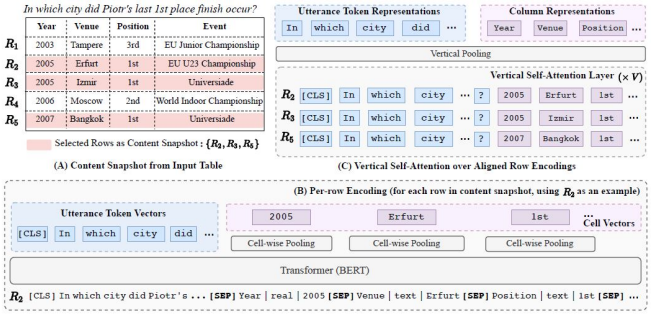

Figure 3: Overview of TABERT for learning representations of utterances and table schemas with an example. (A) A content snapshot of the table is created based on the input natural language utterance. (B) Each row in the snapshot is encoded by a Transformer (only R2 is shown), producing row-wise encodings for utterance tokens and cells. (C) All row-wise encodings are aligned and processed by V vertical self-attention layers, generating utterance and column representations.

For the classification task, we provide two solutions: multiclass classification and binary classification. In the multiclass classification, we determine which column is selected from a table to give the answer. While in the binary classification, we determine each of the columns should be selected or not. The Figure 1 shows the difference between multiclass classification and binary classification. The reason we can apply multiclass classification is that only one column is selected from each table to generate the answer in our dataset. However, to consider the general case that one or more columns can be selected, we also implement binary classification.

We present the detailed framework for classification in Figure 2. As shown in Figure 2, the data is input to the TaBERT encoding layer, which will generate the column encodings and question encodings. We follow the TaBERT convention of data input format for encoding the natural language question together with the headers and cells of the table. Figure 3 shows the procedures of transforming the question and table to question encoding and column encoding by TaBERT with an example, and more details could be found in Yin et al. [2020]. In our model, the size of word embedding for each word in question is 768. We sum the word embeddings of the words in a question as the representation of this question and the size of question encoding is still 768. The size of each column encoding is also 768.

With the output from TaBERT, to leverage the attention info between question and columns to concentrate on important columns, we apply attention layer to column encodings and question encodings. Here we only implement attention layer for multiclass classification. We proposed three ways of applying attention: only applying attention to column encodings(model multi_attention_1); concatenate column encodings and context encoding together as a whole, the apply attention to the concatenated encoding(model multi_attention_2); apply attention to column encodings and context encoding separately(model multi_attention_3). After attention layer, we combine generated column encodings and context encoding as the input to classification layer. We also have two ways of combining them: add them or concatenate them. In the classification layer, it is just traditional classification consisting of two linear layers, activation function and softmax function. The activation functions we have tried are tanh and relu functions.

Aside from column classification, we also conducted row selection, which is an engineering work rather than a machine learning work. We ignore the questions with aggregation, such as Max, Min, Count, etc. We leverage the "conditions" in the dataset. By performing texting cleaning such as removing the punctuation and lowering the cases, we convert the text conditions to selection conditions that computer can execute. Thus, we can select rows that satisfy the conditions. With selected rows and columns, we can finally generate the selected cells.

3.2.2 Sequence to Sequence Model

In this section, we will present of the details Seq2Seq model. As stated in the previous section, the output of the TaBERT encoder is a fixed dimension vector representation, one for the table and one for the context. And we concatenate these two vectors into one vector as the encoder output which will contain

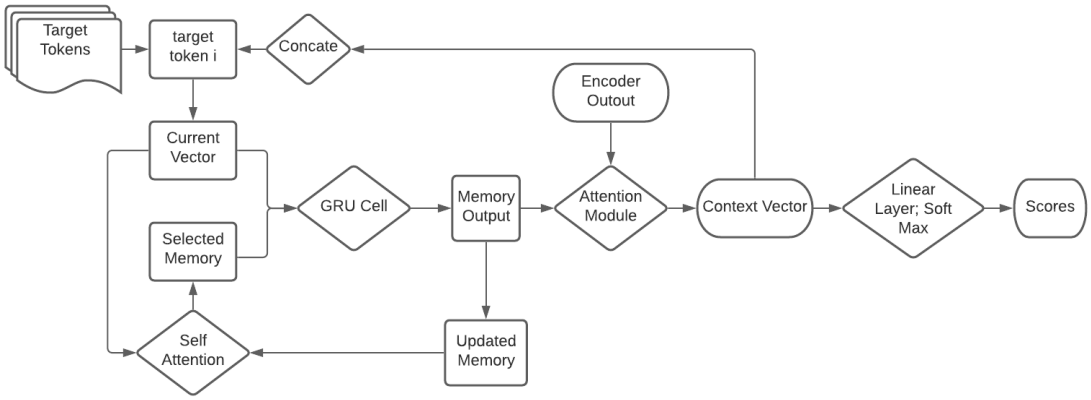

Figure 4: The detailed framework for Seq2Seq Decoder.

the information of both the table and the context. Given we also need a vector representation of the hidden output from the encoder, we first squash the encoder output by summing on the second dimension and then pass it through a linear layer and ReLu.

$$\text{bs:batch size, nc:number of column, src\_len:length of tokenized context}$$

$$\text{table vector shape: } [\text{bs}, \text{nc}, 768], \text{ context vector shape: } [\text{bs}, \text{src\_len}, 768]$$

$$\text{encoder output: } [\text{bs}, \text{src\_len+nc}, 768], \text{ encoder hidden: } [\text{bs}, 768]$$

As for the decoder, the input will be our encoder output, encoder hidden (memory) described above and the target answers. We trimmed the $[eos]$ token off the answer sequence for training, hoping the model will predict $[eos]$ for each sequence.

The nature of Seq2Seq model is recurrent (refer to Figure 4 for detailed flow chart). Thus for the main part of the decoder, it will loop through each vector or token of the target answer. The current vector will be the concatenation of current token and context vector. Context vector is the output from the attention module and it will be explained in more details below.

First we calculate the self attention by pass in the current vector and the hidden output (or memory) into the self-attention module. The calculation will be explained in more details below.

The output of the self-attention and current vector will be passed into GRU recurrent cell and we will have a new memory output (or hidden unit). And this new memory output and encoder output will be passed into attention module. We will call output of it context vector in our case.

Lastly we pass the context vector through a linear layer mapping to the output dimension and apply softmax to obtain the final score.

Before the start of the next loop, we will have to update the memory by incorporating the new memory output from the GRU cell by concatenating on the corresponding dimension; and also update the current vector by concatenating the next token in the target and the new context vector.

Self Attention This idea is first brought up in the paper Attention is All you need by Vaswani et al. [2017], where they introduced the Transformer. In the Seq2Seq language model, the hidden state contains information about previous tokens. The Transformer instead performs attention over all inputs at a given layer. 'Attention' computes an output vector by taking a weighted sum of input vectors. The weights are 'attention weights'. The Transformer uses scaled dot-product attention: The input is query, key and value pair Vaswani et al. [2017]. In our model, the key K is hidden unit (or memory); as for the query Q, we concatenate the current vector with the memory and pass them through drop out, linear layer and another drop out. And the value V is memory, which is the same as the key. And the output will be calculated as below and will be one of the input to the recurrent GRU cell.

$$\text{Attention(Q,K,V)} = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Table:							
Player	No.	Nationality	Position	Years in Toronto	School/Club Team	Question:	
Antoniio Lang	21	United States	Guard-Forward	1999-2000	Duke	Who is the player that wears No 42?	
Voshon Lenard	2	United States	Guard	2002-2003	Minnesota	SQL:	
Martin Lewis	32	United States	Guard-Forward	1996-1997	Butler CC	SELECT Player WHERE No. = 42	
Brad Lohaus	33	United States	Guard-Forward	1996-1996	Iowa	Answer:	
Art Long	42	United States	Guard-Forward	2002-2003	Cincinnati	Art Long	

Figure 5: An example of the WikiSQL semantic parsing dataset. The inputs consist of a table and a question. The outputs consist of a ground truth SQL query and the corresponding result from execution.

phase	table_id	context	sql	answer	sql_query	caption	name	page_id	title	rows	section_title	types	header
1	1-1000181-1	Tell me what the notes are for South Australia	{'sel': 5, 'conds': [[3, 0, 'SOUTH AUSTRALIA']...	[no slogan on current series]	{'agg_index': 0, 'cond_ops': ['=', '>', '<', '...	None	table_1000181_1	NaN	None	[[Australian Capital Territory, blue/white, Ya...	None	[text, text, text, text, text, text]	[[State/territory, text, Australian Capital Te...
1	1-1000181-1	What is the current series where the new serie...	{'sel': 4, 'conds': [[5, 0, 'New series began ...	[cb-06-zz]	{'agg_index': 0, 'cond_ops': ['=', '>', '<', '...	None	table_1000181_1	NaN	None	[[Australian Capital Territory, blue/white, Ya...	None	[text, text, text, text, text, text]	[[State/territory, text, Australian Capital Te...

Figure 6: An example of our dataset. The inputs consist of a table, a question, and corresponding sql query. The outputs consist the selected cells from SQL execution.

Attention Module:

Masked Attention: The following formulas are the snapshot of the calculation in the attention module.

$$\text{Attention\_Score} = \text{batch multiplication(encoder\_output,hidden)}$$

$$\text{Masked\_Attention} = \text{softmax(Attention\_Score*The Source Mask)}$$

$$\text{Context Vector} = \text{Masked\_Attention} * \text{Encoder\_Output}$$

After we get the context vector, we then concatenate Context Vector with hidden and pass it through linear layer and tanh transformation to come up with the final weighted attention Context Vector. Through this calculation, we are hoping to train the model to pay more attention to the column containing the right answer or the token in the question which are more correlated to the final answer.

The Source Mask: The source mask is created by checking where the source sequence is not equal to a <pad> token. It is 1 where the token is not a <pad> token and 0 when it is. And we will transform 0 to some really negative number in the later calculation so the model will pay 0 attention to the <pad> token.

4 Experimental Evaluation

4.1 Data

We use WikiSQL[2] which constructed by Zhong et al. [2017] as our dataset. It is a large natural language to SQL dataset of 80654 hand-annotated examples of questions and SQL queries distributed across 24241 tables from Wikipedia. An example of WikiSQL data is presented in Figure 4. We extract the question, answer, sql query and table content (including headers and rows information) from the WikiSQL dataset to construct our dataset. One table corresponding to several questions with answers in the table. We present an example our dataset in Figure 5.

Before inputting to the TaBERT, we also need to pad the data which enables the tables having the same number of columns. We first diagnose that 99% of tables having less than 15 columns, so we decide the max length of columns as 15 and ignore the tables that have more than 15 columns. Then we pad on table header, content, and label for classification task and padding on answer sequence for attention model. In addition, as explained before in the task section, we remove the questions with aggregation calculation from our dataset to simplify the research problem.

[2] https://github.com/salesforce/WikiSQL

Table 1: fine-tuned models' performance

Model	Learning rate	validation loss	accuracy	f1_score
multi_attention_1	1e-5	2.697 to 2.109	0.7425	0.7426
multi_attention_3	3e-4	2.690 to 1.896	0.9344	0.9345
Seq2Seq	5e-5	10.912 to 2.310	0.2731	0.2043

4.2 Methodology

As our dataset is really big and thus training on the whole sample will take a lot of resources and time, we first randomly choose 100 rows in full train dataset as small sample train data and 20 rows in full validation dataset as small sample validation data to ensure the model converge on the training loss, which implies the model over-fit on the small samples.

With the help of Pytorch lightning modules, we successfully streamline the training code; for example, with the help of pytorch lightning, we remove the for loop of the training epochs, get rid of all the device attaching of tensors and incorporate tensor board to help us track the training loss at ease.

As for training on small samples, we try different ways to apply attentions for multi-class classifications (add or concate question embedding and column embedding) and try different optimizers(SGD and AdamW) and schedulers (ReduceLROnPlateau or LinearSchedulerWarmUp) for both encoder and decoder. To be specific, we use various values of hyper-parameter momentum for optimizer, and various values of hyper-parameter patience and min_lr for scheduler. At the same time, we use Early-Stopping with patience=3 to monitor when the validation loss has stopped decreasing for 3 epochs to prevent over-fitting.

After fine-tuning on the small samples, the multi-class classification model(multi_attention_1, multi_attention_3) and Seq2Seq model converge. In contrast, binary classification and multi_attention_2 do not seem to learn much. Binary classification's final loss is jumping in the general range [0.315,0.320] and it won't decrease any further and for model multi_attention_2, the final loss ends up in the general range of [2.630, 2.634]. Therefore, we focus on multi-class classification model multi_attention_1 and multi_attention_3, and Seq2Seq model for full dataset.

As for training on full dataset, we train our models on GPU, and it takes about two days for each model to complete the training. We use configuration including optimizer SGD with momentum 0.99, scheduler ReduceLROnPlateauworks with patience 5 and min_lr=7.5e-5; additionally concating question embedding and column embedding for multi-class classification works the best compared to other methods like summation.

4.3 Results

We use accuracy and f1-score as our evaluation metrics. It is hard to compute f1-scores for multi-class classifications without module. Due to time efficiency, for multi-class classifications, instead of calculating f1-score for all the predictions and true values, we calculate the f1-score for each batch and take the average of these scores. Table 1 shows the fine-tuned models' validation loss and accuracy.

From the table, we can see that in general, classification model has much better performance than seq-to-seq model in terms of accuracy and f1 score. The best performing model is multi_attention_3 , which is multi-class classification model with attention applying to column encoding and context encoding separately, and concatenation of question embedding and column embedding. In terms of converge of validation loss, multi_attention_3 also has the lowest validation loss after training and larger accuracy and f1 score. However, Seq2Seq model's performances are not as promising as the classification model with low accuracy and f1 score. The differences in accuracy between models are statistically significant as the accuracy for multi_attention_3 is about 5 times than that of seq-to-seq.

4.4 Discussion

In general, the high accuracy of multi-classification model supports the hypothesis that classification is more suitable for the question-answering task on tabular data, while the bad performance of seq-to-seq model implies that it has some shortcomings to deal with this problem. In general, the way of grabbing information for classification model is similar to human strategy to tackle this task, as people also first search for the needed column and then find the exact row to get the position of the cell. While by directly

looking at the whole table as what seq-to-seq model does, it is more challenging for the model or human to generate the right answer.

Among all the classification problem we have tried, the reason that the binary classification does not converge may due to imbalanced numbers of 0s and 1s since there are 5 times more 0 than 1. Even though we add weight to cross entropy loss layer, the model still faces problems on generating useful information. As multi-class classification model only has one number to indicate the correct column, it does not have this kind of problem. Moreover, different ways of applying attention in the multi-class classification decoder also affect the performances.

Deep dive into the result for the multi-classification, multi_attention_2 does not learn things well as the validation loss does not converge, and model multi_attention_1 learn something while not as much as model multi_attention_3. The following are some of the reasoning behind this:

As multi_attention_1 applies attention to only column encoding, and model multi_attention_3 applies attention to column and context encoding separately; this implies that adding attention to the context encoding will improve the performances.

However, multi_attention_2 where it applies attention to the concatenation of table and context encoding performs the worst, implies that attention to the combination surprisingly brings extra information that hurt the model performances.

As for the Seq2Seq model, we conduct error analysis on the Seq2Seq Model and print out some of the error answers. The model performs the worst when answers contain numbers and characters, for example '8 cakes'. We conduct some research by reading other papers on this matter, the performances of classification always work better than the Seq2Seq model. Unlike question and answering for a document, the similarity between rows or columns is rather smaller than in a document and by passing all the table and context information into the Seq2Seq model makes it more challenging to generate the true answer.

As for the final production, our model only performs on the matter of cell selection , i.e. without aggregation. In the real life scenarios, there will be more cases like summation, mean, rank and other aggregation task in this area of research and further work is needed to incorporate more complicated scenarios like this to make the model more comprehensive.

5 Conclusions

This project tries to implement TaBERT as the encoder to transform the natural language questions on the tabular data to answers without the logic form like sql. We have finished the column classification task by implementing the binary classification and mutli-class classification with three different ways of attentions. Besides, we also implemented the sequence-to-sequence GRU model with self-attention. The accuracy for multi_attention_1 model is 0.7425, for multi_attention_3 model is 0.9344, and for Seq2Seq model is 0.2731. The multi-class classification performs much better than Seq2Seq model.

For future work, finding another model structure other than classification and seq-to-seq as decoder and comparing the performances is an interesting field. In addition, we currently ignore all the aggregation questions, in the future, we could develop more complex structure to incorporate this part as applying different aggregation operator on top of the selection of multiply rows.

6 Lessons learned

From this project, we implement NLP model such as state of the art BERT language model, which we learn from the Class, into real research problem. Unlike the class project we have done, this project contains larger and messier data and more complicated model structure.

Given the complex nature, our mentor taught us to form strategy to breakdown complicated problem into pieces, like building simple model at first to test the performance and increase the complexity and difficulty gradually. Also, we have the chance to learn some of the most recent development in the filed of NLP like TaBERT and TAPAS, which are just released earlier this year by Facebook and Google, and be able to elaborate their result in our project.

In addition to learn building model on complex task and input, we also practice to use different methods and strategy to let model converge as in real world application. Training a large neural network is more

than a science but rather an art. Different hidden dimensions, learning rates, optimizers, loss schedulers, patience, early stopping all play vital role in the training process. Though the process is painful and lengthy but there are a lot of take away that we will definitely use in our future projects. For example, in order to save the training time and resources, we will first try to focus on one hyper-parameter to train and optimize it to a optimal range and then use this setting and move on to the next parameter; also training on smaller samples to make the model overfit also help us speed up the training process. Last but not least, the error analysis of this model help us to understand the true performance of the neural network which works like a black box. These negative examples help us to understand the scenario where the model fails to predict, which will give us better idea for future model improvements.

7 Student contributions

JG+JZ+YW: wrote report,

YW: implemented SeqtoSeq GRU model,

JG+JZ implemented column classification and row selection.

References

Tong Guo and Huilin Gao. Table2answer: Read the database and answer without sql, 2019.

Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. Tapas: Weakly supervised table parsing via pre-training, 2020.

Pengcheng Yin, Graham Neubig, Wen tau Yih, and Sebastian Riedel. Tabert: Pretraining for joint understanding of textual and tabular data, 2020.

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.