Natural Language Models for Data Visualization Utilizing nvBench Dataset

Shuo Wang and Carlos Crespo-Quinones

DATASCI 266: Natural Language Processing
UC Berkeley School of Information
{shuo.wang, carlos.d.crespo}@ischool.berkeley.edu

Abstract

Translation of natural language into syntactically correct commands for data visualization is an important application of natural language models and could be leveraged to many different tasks. A closely related effort is the task of translating natural languages into sql queries, which in turn could be translated into visualization with additional information from the natural language query supplied[1]. In order to contribute to the progress in this area of research, we built natural language translation models to construct simplified versions of data and visualization queries in a syntax called Vega Zero first proposed by Luo, Yuyu, et al[2]. In this paper, we explore the design and performance of these sequence to sequence transformer based machine learning architecture using large language models such as BERT as encoders to predict visualization commands from natural language queries.

Introduction

Data visualization is an integral part of data analysis and communcation, highly skilled professionals spend significant portion of their working hours constructing data queries and turn them into charts and graphs. The potential application of machine learning models to understand the underlying data and translate natural language queries into data visualization would greatly enhance the efficiency and productivity of many tasks. Not only would such tools enhance our existing ability to understand and communicate our data, it could also help us uncover previously overlooked information.

Some immediate applications include: creating bar, line and scatter plots of sql tables, transform and group data with a table and display the aggregated data and joining and combining tables of data to extract information and display visually.

Previous research in this area has included explorations of transforming natural language into SQL commands[1, 3] with deep learning models and natural language to visualization interfaces[4]. NvBench is a new dataset created to facilitate the research to further integrate the process of natural language to data query and data query to visualizations[2] . Leveraging this data set and the ncNet transformer model architecture[5], we further explore the various modeling possibilty in our work.

In this paper, we have used the nvBench as our train, validation and test dataset to create BERT encoder based multi-head attention transformer models and compare the performance of the new models with the performance of the ncNet model (also a sequence to sequence transformer model). Our goal is to explore the possibility of a generalized natural language to visualization process which the ability of the system to handle natural language inputs is not limited by the input training data. We have provided below the main results of our research and detailed description and analysis of the input data, model architecture and model performance.

From our reseach, the BERT encoder based sequence to sequence transformer model was able to achieve an overall accuracy of 79%, confirming our hypothesis of the possibility to leverage transferred learning of BERT models to data visualization tasks.

Background

Natural language to visualization research has been conducted for decades, some of these efforts include early approaches to manually program rule-based logic for the handling of anticipated user inputs in multi-model systems[4] and more recent attempts to incorporate machine learning and optimization techniques[6]. However these systems are limited in functionality by the ability of the designer of the systems to anticipate the possible inputs from users. What if the user uses a word that the system has never seen? What if the natural language input was written in a idiomatic way that was not anticipated by the training data? These limitations could potentially be addressed with the recent advances in large language models such as BERT and GPT through transfer learning.

Recent research exists for applying BERT model to the task of data visualization[7] by converting natural language into vector representation embeddings and

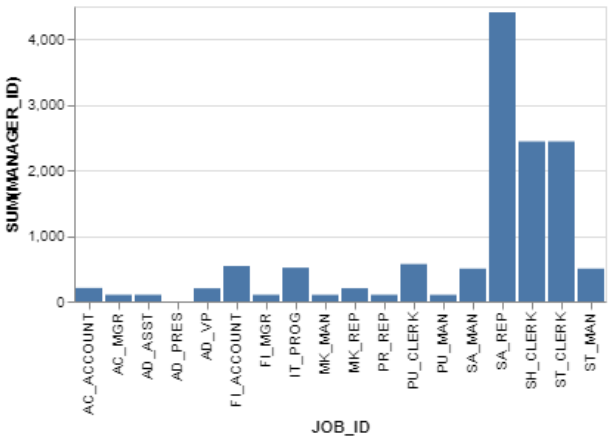

Figure 1: *Rendering: Visualize BAR SELECT JOB_ID , SUM(MANAGER_ID) FROM employees WHERE salary BETWEEN 8000 AND 12000 AND commission_pct != "null" OR department_id != 40 GROUP BY JOB_ID ORDER BY JOB_ID ASC.*

use these embeddings to predict various ingredients needed for the visualization of tabular data such as chart type, data columns and aggregation type. However, the structured approach to the problem inherently limits the expressive power of the system to generate complex visualizations.

A new dataset, nvBench[2], was made publicly available for research purposes in 2021. The dataset contains pairs of natural language queries and data visualization commands in vega zero syntax[5]. A self-contained sequence-to-sequence model, ncNet[5], created for and trained on the dataset was also made available. We base our research on this dataset and the ncNet transformer model, modifying the architecture to use BERT and other language models to train and test generalizable natural lanuage to data visualization models.

Methods

Data

nvBench The dataset contains 7247 visualization queries (labels). Each query contains visualization type and data retrieval command meant to be run against an underlying database with multiple tables, which contain the actual data to be display. Below is an example of a visualization query:

> Visualize BAR SELECT JOB_ID , SUM(MANAGER_ID) FROM employees WHERE salary BETWEEN 8000 AND 12000 AND commission_pct != "null" OR department_id != 40 GROUP BY JOB_ID ORDER BY JOB_ID ASC

In this example, the chart type is “PIE”, followed by the SQL statement for data retrieval. Figure 1 shows the rendering result of the query.

The example visualization query is mapped to several natural language queries, from which we train our models:

> 1. For those employees whose salary is in the range of 8000 and 12000 and commission is not null or department number does not equal to 40, show me about the distribution of job_id and the sum of manager_id , and group by attribute job_id in a bar chart, I want to sort in ascending by the bar.
>
> 2. For those employees whose salary is in the range of 8000 and 12000 and commission is not null or department number does not equal to 40, a bar chart shows the distribution of job_id and the sum of manager_id , and group by attribute job_id, and could you order x axis in asc order?

In total, there are 25762 natural language queries, averaging 3.55 natural langauge queries for every visualization query.

Augmented Data Due to the complexity of SQL syntax, a simplified version of the visualization query is used in place of the original query, called vega zero[5]:

> mark bar data employees encoding x hire_date y aggregate count hire_date transform filter salary between 8000 and 12000 and commission_pct != "null" or department_id != 40 bin x by month

Where the visualization query is flattened into a sentence with special markers designating locations of information, please refer to [5] for complete syntax. In our research, we follow the same syntax rules.

Since this syntax only supports single table queries, it is not possible to predict queries where the joining of tables are necessary, so we only train and test our models on a subset of nvBench dataset, with 2988 visualization queries for training, 186 visualization queries for validation and 625 visualization queries for test.

The natural language queries are also augmented to include the template of vega zero queries:

> <N> For those employees whose salary is in the range of 8000 and 12000 and commission is not null or department number does not equal to 40 , draw a line chart about the change of department_id over hire_date , display by the X from low to high . </N>

<C> mark [T] data employees encoding x [X] y aggregate [AggFunction] [Y] color [Z] transform filter [F] group [G] bin [B] sort [S] topk [K] </C> <D> employees <COL> hire_date salary department_id last_name first_name </COL> <VAL> Bull Lex Seo Bell Chen Lee Gee Banda King Baer Fay </VAL> </D>

The natural language query is enclosed in <N> </N> markers, the vega zero template is enclosed in <C> </D> markers and the locations of vega zero query fields are marked by special tokens: [T], [X], [Y], [Z], [F], [G], [B], [S] and [K]. These are the syntax used by the original ncNet sequence to sequence transformer model and we follow the same rules for our models as well.

Finally, every natural language and visualization query pair in the dataset is augmented with two versions, one with the chart type kept as placeholder, [T], and another with the chart type filled in (bar, line, etc). The motivation is to train the model to predict the visualization query whether the chart type is explicitly specified or not.

Data Loader The augmented data were provided in the Github repo for the ncNet paper[5] , which we use unaltered. However, the original dataloader no longer works the latest version of Pytorch, we rewrote the dataloader to be used for training and testing the ncNet model and our own models.

Model

ncNet The ncNet[5] model is a sequence to sequence multi-head transformer model, where the encoder transforms the natural language queries into embeddings combining position and token type information, and apply multiple layers of multi-head attention transformers to incorporate contextual information into the encoded embeddings. Shown in Figure 2, The decoder then takes in the encoded embeddings and the embeddings of the previously predicted tokens to further transform the predicted tokens with attention to the encoded embeddings. We us the ncNet model as our baseline, and replace the encoder with various different architectures.

One important fact to note is that the ncNet model uses all words occuring in the training, testing and validation dataset as vocabulary. Therefore letting the possibility of natural language query inputs.

We also remove the attention mask (referred to as attention forcing in Luo 2022[5]) used in ncNet to create another baseline for our experiments, since attention mask are not applicable to our models.

BERT Model We then proceed to replace the ncNet encoder with the BERT model encoder, as shown in Figure 3. Because the BERT model is pre-trained with a much larger vocabulary, the hope is that this knowledge would transfer to the new model, ncBERT, once we fine tune it. We train an additional BERT encoder model with convolutional layers added to the BERT embeddings, in order to distill relevant information from BERT model embeddings before decoding, as the BERT embedding has a much larger dimension that may contain information irrelevant to the data visualization task. The convolutional layers incorporated model is shown in Figure 4 .

Combining ncNet and BERT Model Finally, we create an encoder combining both the ncNet and BERT encoders, we achieve this by reshaping the embeddings from BERT to the same dimension as ncNet encoder and concatenating them together. The architecture of the model can be found in Figure 5. The motivation of creating this model is to analyze the effect of BERT embedding on the baseline model and explore the potential of enhancing the original encoder without completely replacing it. However, the implication of this approach is that words that does not exist in the original ncNet encoder still could not be used, further research needs to be done to remove this limitation.

Evaluation

The models are trained and tested with training dataset of size 25238, validation dataset of size 1430 and test dataset of size 4920. Each model is run over at least 5 epochs with a learning rate of 0.0005 and the resulting losses are recorded.

Then the accuracy of the models are evaluated by running predictions with the models over the test dataset, with the natural language query tokens and the first n-1 tokens of the label as input (the label tokens are masked so that successive predictions of label are not affected by future label tokens), counting the total number of correct predictions and dividing it by the total number of predictions.

Finally, we evaluate the models one more time with a guide search algoirthm to prediction. The algorithm starts the prediction with the start of sentence token, then repeatedly predicts the next label token with guidance until the end of sentence token is reached. This evaluation provides a more realistic measure of the model's usability. The accuracy from this evaluation is defined as the total number of complete label match over the total number of test labels.

Table 1: *Model Test Accuracy.*

Model	Query	Query+Chart	Overall
ncNet	0.952	0.960	0.956
ncNet w/o AF	0.954	0.961	0.957
nvBERT	0.788	0.788	0.788
nvBERT_CNN	0.788	0.788	0.788
nvncNetBERT	0.887	0.893	0.890

Table 2: *Model Test Accuracy with Guided Search.*

Model	Query	Query+Chart	Overall
ncNet	0.622	0.723	0.673
ncNet w/o AF	0.650	0.745	0.698
nvBERT	0.0252	0.131	0.0780
nvBERT_CNN	0.0195	0.119	0.0693
nvncNetBERT	0.211	0.334	0.272

Results

Accuracy

After running 5 epochs with learning rate 0.0005 and keeping the model version with the lowest validation loss, the test accuracy of the models are reported in Table 1. The accuracy of a model is defined as the probability of predicting the next word correctly given part of the label query.

The table shows three columns, the accuracy of the models on queries where chart type is not specified (Query), the accuracy of the models on queries where chart type is specified (Query+Chart) and the overall accuracy (Overall).

As we can see from the results, the origina ncNet models produced test accuracies in excess of 95%. While the BERT based transformer models were able to achieve close to 80% accuracy. This is attributable to the fact that the BERT model contains a much larger vocabulary than the ncNet model trained only on the vocabulary of tokens present in the dataset.

Further more, the removal of attention forcing in the ncNet model does not appear to affect the test accuracy, suggesting that its absence in nvBERT models had no bearing on the accuracy results of the nvBET models. The accuracies of nvBERT models with and without convolutional layers does not appear to affect the accuracy at all, suggesting minimal impact of the convolutional layers.

Finally the combined ncNet and BERT encoder produced results that improved unpon the nvBERT only models, suggesting that were a model able to combine the ncNET and BERT encoder while remaining generalizable, the results would improve along with the benefits of transfer learning.

Besides the overall accuracy difference, the general trend is that the accuracy for query only tests are slightly lower than accuracy for query and chart type tests. This is expected since the query only tests require the model to predict an additional field. However, given that the accuracy differences are small, the chart type prediction is generally accurate.

Guided Search Accuracy

We next look at the accuracy of predicting the entire label query. The algorithm of guided search, as used by the original ncNet model, proceeds as follows: starting with the start of sentence token for the label, successively predict the top five candidates. Within the top five candidates, the top candidate is chosen unless it does not belong to the possible words for the current position being predicted, in which case the rest of the candidates are iterated over to find a suitable choice. Table 2 shows the results for each model.

From looking at the results, it's immediately clear that the lower accuracy in the BERT based models are amplified through the guided search process, resulting in a low overall accuracy of 7.8%. This effect also shows up in the difference between query only and query with chart type accuracy, where the small difference in Table 1 becomes much larger in Table 2.

Training and Loss

Show the training and loss history...

Sample analysis

Show some examples of correct and incorrect predictions...

Discussion

Discuss how our tests demonstrated the potential of using BERT and other generalized language model for this problem...

References

[1] Victor. Zhong et al. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv:1709.00103v7 [cs.CL]*, November 2017.

[2] Yuyu. Luo et al. nvbench: A large-scale synthesized dataset for cross-domain natural language

to visualization task. *arXiv:2112.12926v1 [cs.HC]*, December 2021.

[3] Tao. Yu et al. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. *arXiv:1909.05378v1 [cs.CL]*, September 2019.

[4] Kenneth. Cox et al. A multi-modal natural language interface to an information visualization environment. *International Journal of Speech Technology*, 2001.

[5] Yuyu. Luo et al. Natural language to visualization by neural machine translation. *IEEE Transactions on Visualization and Computer Graphics, vol. 28, no. 1, pp. 217-226*, January 2022.

[6] Jillian. Aurisano et al. Articulate2: Toward a conversational interface for visual data exploration. *IEEE VIS '16 (Poster paper)*, January 2016.

[7] Can. Liu et al. Advisor: Automatic visualization answer for natural-language question on tabular data. *2021 IEEE 14th Pacific Visualization Symposium (PacificVis)*, January 2021.

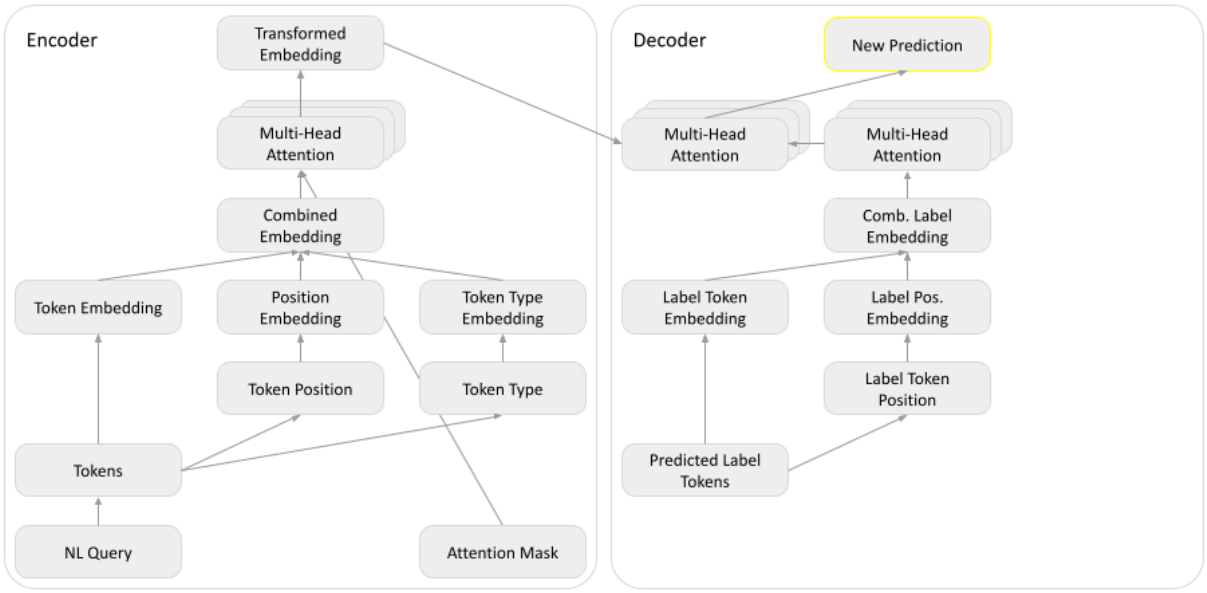

Figure 2: *Architecture of ncNet model.*

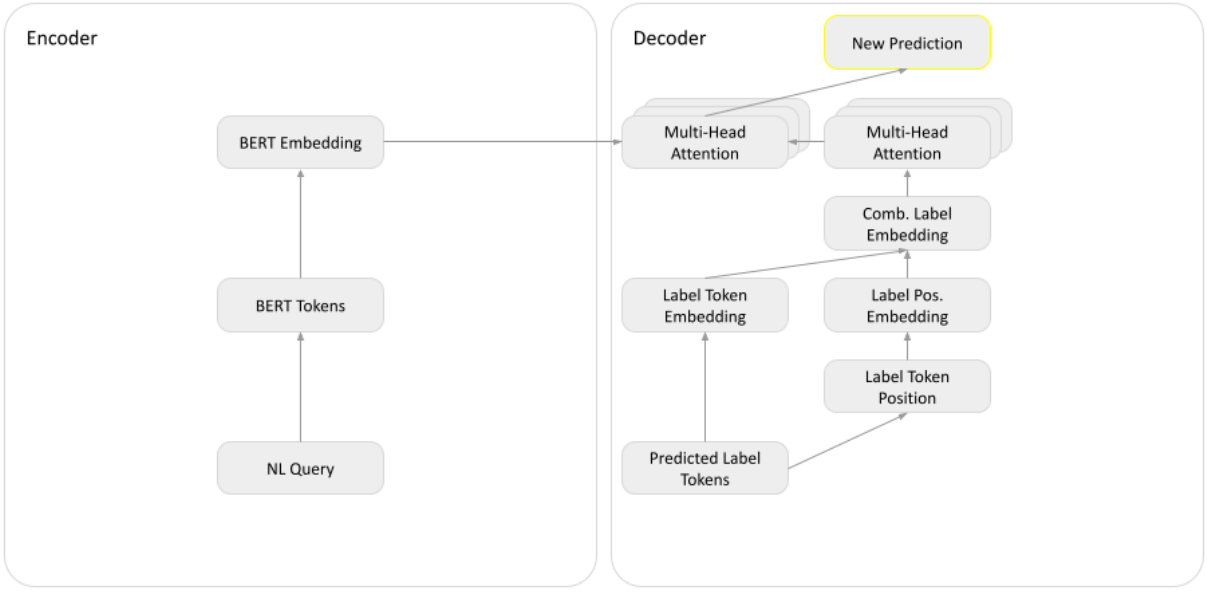

Figure 3: *Architecture of ncBERT model.*

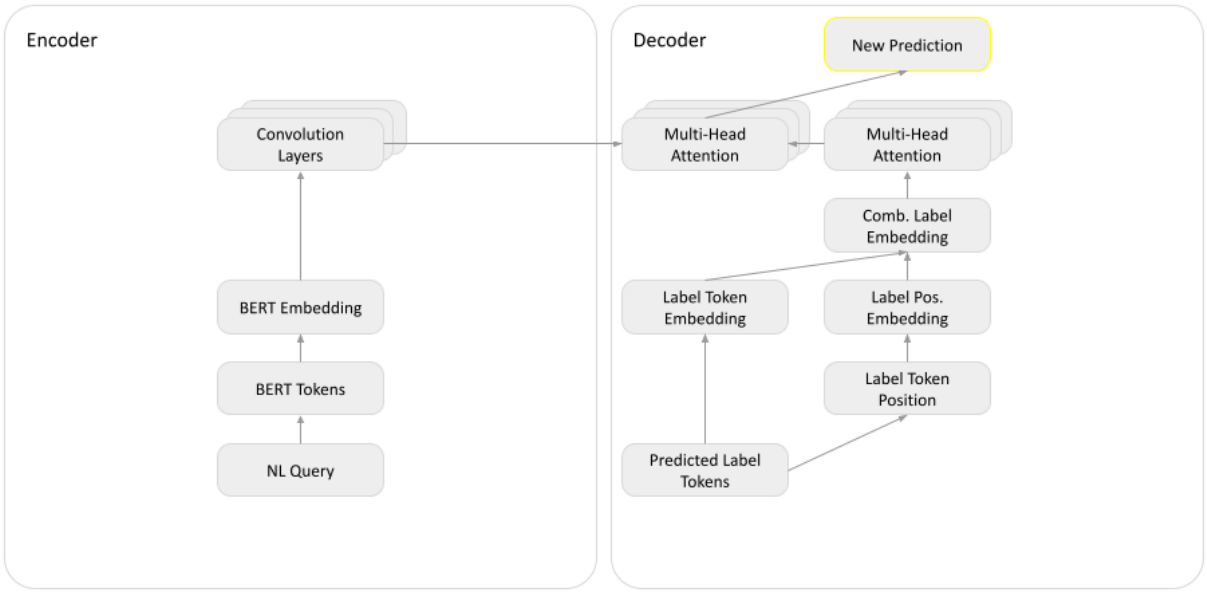

Figure 4: *Architecture of ncBERT_CNN model.*

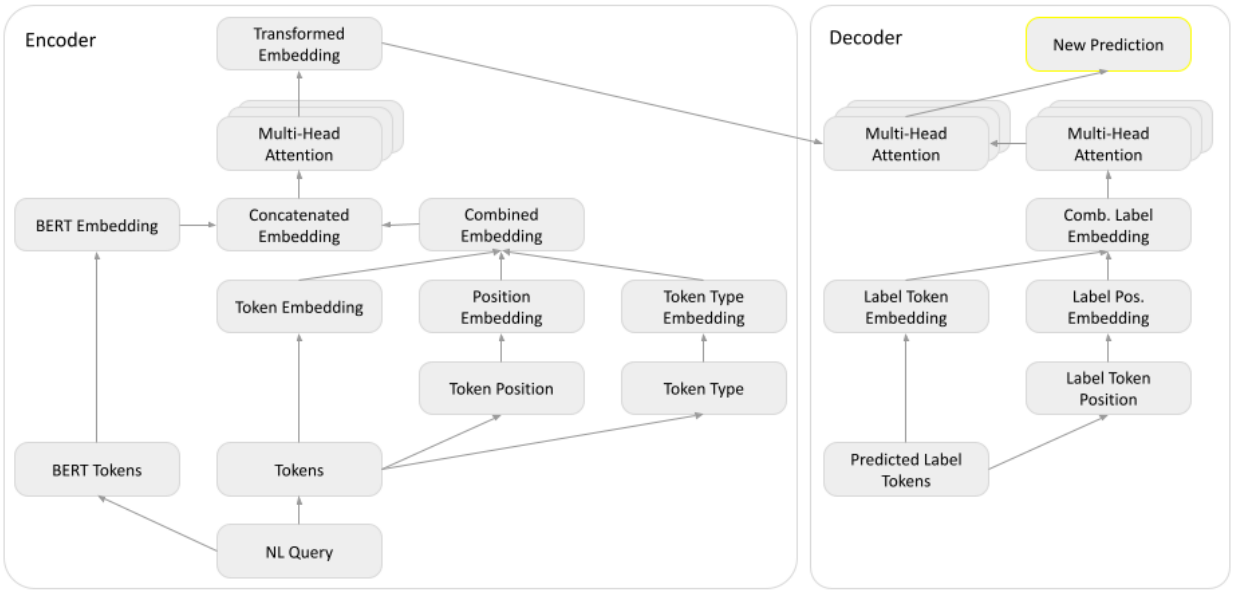

Figure 5: *Architecture of ncNetBERT model.*