Akash Krishna
Final Paper

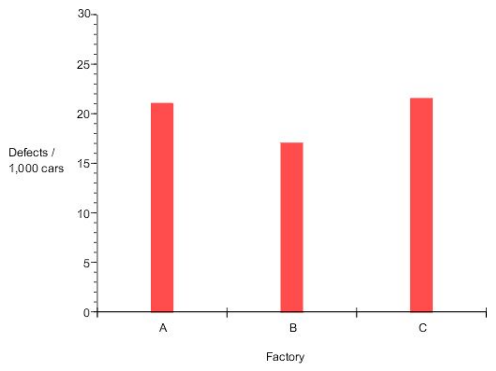
Question Answering System for Data Visualizations

Introduction:

Building accessible data visualizations has been a topic of conversation for a while now. There exist web-based standards as outlined in the Web Content Accessibility Guidelines. Because of these standards, companies have had to come out with solutions to make data visualizations accessible on the web. Most of these solutions involve embedding data within the HTML elements that make up the figure so that a screen reader will be able to read out the data when going through the HTML elements of the figure. There are automatic techniques to embed data into the HTML elements of a data visualization. At a start-up called Fizz Studio, they developed a charting package that outputs an accessible graphic when given the data of the graph.

Unlike content on the web, there are no mandated guidelines for accessibility when it comes to PDF formatted files, which is a big issue since the common format of research papers is PDF. Finding a way to make PDF files accessible is very important because research papers often use figures for reporting quantitative results and analysis. Without methods of making the content in the figures accessible, those with visual impairments are not able to understand the content of the figures in these papers. This is a significant hurdle those with visual impairments have to overcome when doing work in higher education. There are ways to make an accessible PDF file, but the process is an extremely time-intensive one; there is no automated procedure similar to making accessible HTML-based figures. The current process of making a PDF accessible is to manually add tags to a PDF document by using an application like Adobe Acrobat, which can be excessively laborious especially if you wanted to add a tag to every data point on a scatterplot or other types of data visualizations. Simply adding a caption for a figure does not tell the whole story of the content within a figure, so it is important to tag all the components of a figure to provide those with visual impairments the ability to have access to the same content as those without visual impairments (Bigham et al. 2016).

The goal of my project is to create a question answering system to query information about a data visualization given an image of a figure. By being able to query information from a figure, those with visual impairments will be able to access most of the same information a visually able person can attain by looking at the figure.



Given an image such as the one on the right, the program will allow the user to input natural language questions such as: “what is the x-axis title?” and the output will be “Factory,” or “What is the first x-axis label?” and the output will be “A,” or “what is the value when the factory is A?” and the output will be “21.” Through this means, someone with visual impairments will be able to understand the content within a data visualization.

Such a task will involve two components: data extraction from a figure and natural language processing of the inputted question.

Related Work

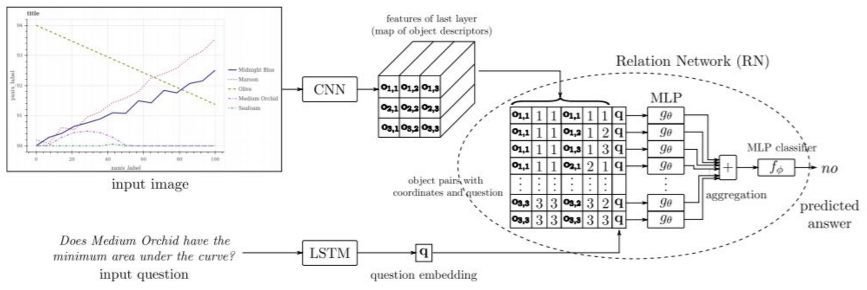
The task of providing a question answering system for data visualizations has been done in the in two papers: FigureQA (Kahou et al. 2018) and FigureSeer (Siegal et al. 2016).

The goal of FigureQA was to develop models to understand patterns in visual representations of data. So, the task was the train a model to answer a natural language query based on the inputted image of a data visualization without extracting the data from the figure. FigureQA utilized a synthetic corpus of images. To generate the figures, numerical data was sampled according to a set of constraints designed to make sampled figures appear natural. Then, Bokeh open-source plotting library was used to plot the data. For each figure generated, there were associated question-answer pairs where the questions concern various relationships between plot elements and examine characteristics like the maximum, the minimum, area-under-the-curve, smoothness, and intersection. Each question was a binary question with a yes or no answer.

The model utilized in the FigureQA paper was a Relation Network (RN). RNs are a neural network module built for relational reasoning. The composite function of the RN is given in the image on the right.

$$RN(\mathbf{O}) = f_\phi\left(\frac{1}{N^2}\sum_{i,j} g_\theta(\mathbf{o}_{i,\cdot}, \mathbf{o}_{j,\cdot})\right)$$

Each o_i and o_j are object representations that are each fed into an MLP, g_θ. The output from each g_θ is then fed into another MLP, f_θ, to get the final output. Each pixel of the CNN output corresponds to one “object” o_i. You then concatenate the location of the pixel in the CNN output with an LSTM model representation of the natural language query to create the object representations. Then you can take all the object representations and run



them through the relational network to get the output.

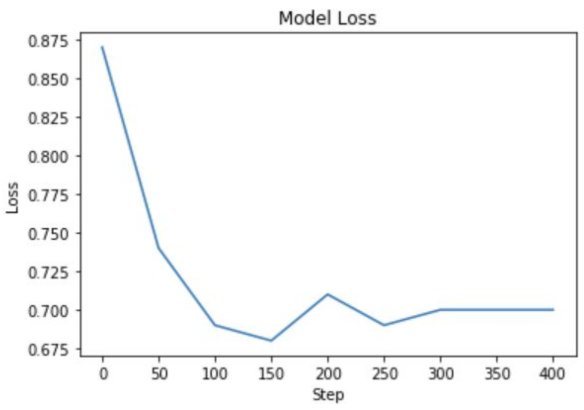
The work of FigureSeer was targeted towards allowing search engines to access the content within a figure in a research paper. Unlike FigureQA, FigureSeer first attempts to extract the data from the figure and then uses that data to answer a natural language query.

For the figure classification and extraction portion, a CNN is used to classify the type of image—whether it was a bar graph, pie chart, etc—and then Optical Character Recognition tools are used to parse the text in the image. It uses techniques like finding areas with vertical and horizontal lines of text to find the axes, and similar techniques to locate the legend. Then, to parse the data, it uses optimization functions to follow the path of the data and extract the locations of the data points.

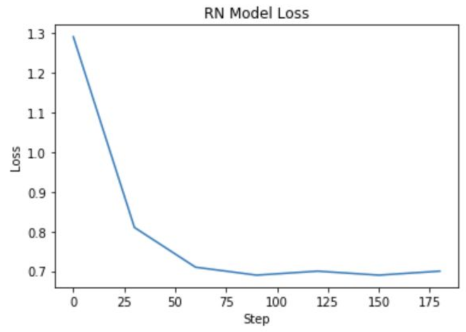
For the query answering portion, the program expects a templated query where you specify the axis title and labels. Then using the parsed data, the program returns the value at the specified location on the axis provided.

Pre-Midterm:

I ran a simpler model compared to the Relation Network used in the FigureQA paper. This model involves concatenating the output of the CNN with the output of the LSTM and running that through an MLP to get a final yes or no answer to the input question. Running this model on a quarter of the dataset, I have achieved 55% accuracy, which is insignificant considering it was for answering binary questions. However, this mark seems to match the accuracy achieved in the FigureQA paper for the same model. The figure on the right shows the loss for the CNN + LSTM model.



I also ran the FigureQA's Relation Network model on a tenth of the dataset and achieved an accuracy of 58% which is far below the accuracy achieved in the paper of 72%, but is still better than the accuracy of the simplified CNN+LSTM model.



The code I used came from FigureQA's github repository: https://github.com/vmichals/FigureQA-baseline with some minor modifications I added to allow it to work on my machine and to use updated modules.

Dataset:

Dataset 1:

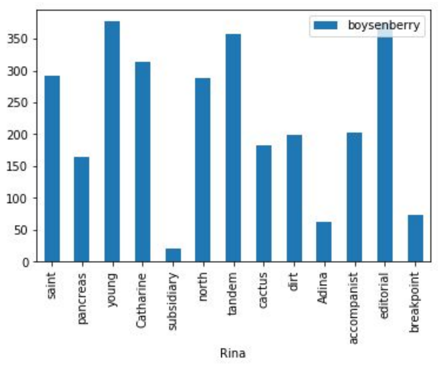
I spoke with the founder of an accessibility start-up called Fizz Studio, and he told me that there is essentially a three step process people take to interpret a data visualization. First, they seek to understand what the visualization is about. Then, they look to query data. Lastly, they look to perform computation on the data. Those who are visually-able perform such steps simply by looking at a visualization. Those that have visual impairments seek the surrounding text to get context on the visualization, but there is no way for them to query data from the figure and then be able to compute statistical operations like the mean, median, mode, range, etc. There was no dataset available with question answer pairs for querying information from a figure, so I created my own datasets.
The first dataset I created tags a question to its type. There are three possible types: getting context (Type 0), querying data (Type 1), and performing operations (Type 2). So the first dataset had question and type pairs like:

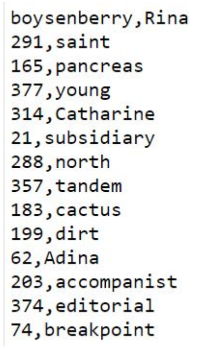
What are the first five x-axis labels?:Type0
Tell me the y-axis title:Type0
What is the precipitation when the month is February:Type1
What is the value for February?:Type1
What is the max value?:Type2
Which month is the median?:Type2

I created this dataset by randomizing the question structure, putting random nouns within the sentence, randomizing the value being queried, randomizing the operation in the question, etc. The intuition behind creating this dataset is that each question type has a different set of computations required to return the correct output. Type 0 questions require looking at the axis titles and the axis labels, Type 1 questions require looking at the data extracted, and Type 2 questions require performing a computation on the data. So, if I am able to accurately classify the type of question, I would be able to perform the right computation for the given question.

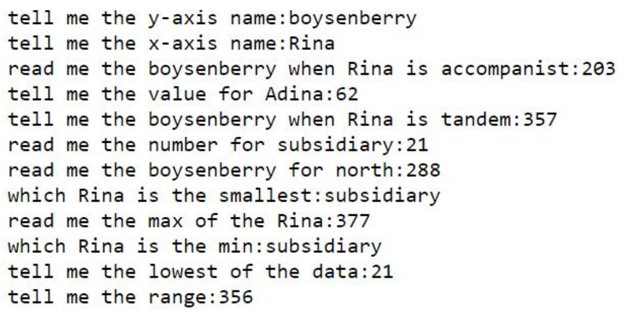
Dataset 2:

The second dataset I created is a dataset with a barchart, its respective data, and the question-answer pairs associated with that chart. I focused on barcharts because I didn't have time to code the image processing techniques necessary to extract information for other types of visualizations. This dataset has 100 charts, 100 tables, and 1500 question-answer pairs (15 questions for each chart). The size of this dataset can easily be scaled up by changing a parameter value in my code from 100 to whatever value needed. I am using this dataset to extract data from the chart and then use that to answer the natural language query.





```
boysenberry,Rina
291,saint
165,pancreas
377,young
314,Catharine
21,subsidiary
288,north
357,tandem
183,cactus
199,dirt
62,Adina
203,accompanist
374,editorial
74,breakpoint
```



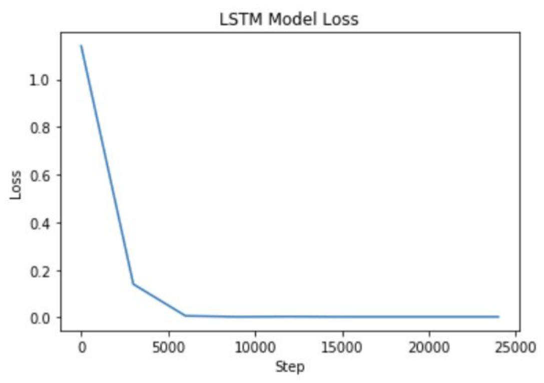
```
tell me the y-axis name:boysenberry
tell me the x-axis name:Rina
read me the boysenberry when Rina is accompanist:203
tell me the value for Adina:62
tell me the boysenberry when Rina is tandem:357
read me the number for subsidiary:21
read me the boysenberry for north:288
which Rina is the smallest:subsidiary
read me the max of the Rina:377
which Rina is the min:subsidiary
tell me the lowest of the data:21
tell me the range:356
```

The above images display the chart, csv formatted data, and the question-answer pairs. The labels and the titles are all randomized, so the data may seem like nonsense.

There are a few limitations on the data. I have restricted the data to contain 20 data points maximum. Type 1 and type 2 questions can only ask for 1 value. However, for type 0 questions regarding the context, users can query multiple labels by asking something like: "What are the first three x-axis labels?" Also, it is assumed that the input image will be a typical vertical barchart where the x-axis is the horizontal axis and the y-axis is the vertical axis.

Model:

I created an LSTM model to classify the question with its associated type. To train this model, I am using my first dataset of question and type pairs. I am using the 50 dimensional Glove word embedding. This model ultimately achieved 99% accuracy in classifying the question with its associated type. However, because I am generating the data, the high accuracy could be the byproduct of a lack of



diversity in the questions. The image on the right displays the loss for the model.

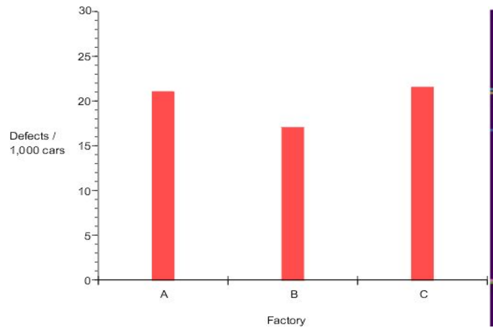
After the question is classified, I am utilizing part-of-speech tagging to extract out the adjectives and the nouns from the question. Then, depending on the type of the question, I perform some computation to determine the output value. For example, if the question is classified as type 0, and the question has either “x-axis” or “horizontal” in the question, then I know that the possible domain of responses are the x-axis labels and the x-axis title. Then, I can look to see if they are trying to query the title or a label and if label, which label(s). Similarly, if the question is classified as type 2, then I know to look for an operation in the question. If a word in the question matches a known list operations I have stored, then I perform that operation on the data. If no known operation is specified, I am using cosine distance to see which operation is the closest to the operation specified in the question and performing that operation.

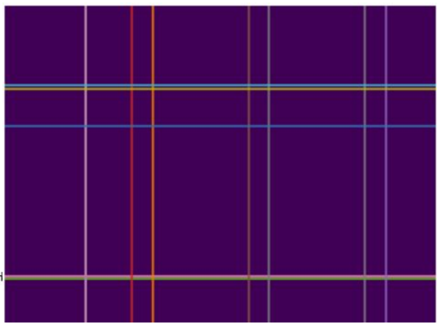
Overall, without including the image processing portion of the project, the code I have implemented works with 97% accuracy. Again, this could be inflated due to the lack of diversity in my questions and because of the lack of complexity of the questions. The reason it does not achieve 100% accuracy is because the part of speech tagger sometimes does not tag the correct part of speech to the word, so it interferes with the computation.

Image Processing:

Currently, I am utilizing an image processing technique called the Hough transform to extract data from a barchart image. The Hough transform works by finding shapes with a small number of parameters like lines, which only have two parameters: a slope and an intercept. Using the Hough Transform, I can extract all the horizontal and vertical lines from a barchart image. The leftmost vertical line is assumed to be the y-axis and the bottom-most horizontal line is assumed to be the x-axis. The intersection of these lines is the origin. Once I have mapped out the axes, The first two vertical lines represent the first bar, the next two represent the second bar, etc. Depending on where the bars are located on the x-axis, I map them to a corresponding x-axis label. Similarly, for each bar, I extend the top-most vertical line to the y-axis and provide the bar a y-value depending on the extracted y-values.

The image on the left gets converted into the image on the right following the Hough transform and after

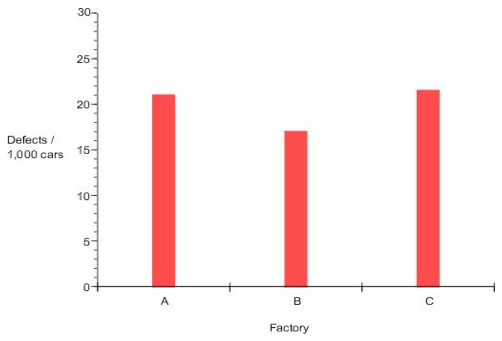




extending the lines. There are four horizontal lines, 3 for the tops of the bars and one for the x-axis. There are 7 vertical lines, 6 lines for the three bars and one for the y-axis

In order to extract the text from the image, I am using Python-tesseract's OCR (Optical Character Recognition) tool. This is how I get the labels and the titles from the chart. However, the OCR tool is very inaccurate especially since the labels are fairly small.

For example, for the image on the right, the output y-axis labels were 25, 10, 5, and 0, and the output x-axis labels were A and B. So, the text extraction missed two labels, one from the y-axis and one from the x-axis. This is one of the better outputs. As the label size gets longer, the OCR is even less accurate.



The accuracy of the whole process from the image processing to the question answering is significantly low, close to 0% because it is not able to extract all the data. So, performing operations will lead to incorrect values, questions like those asking for the third axis label will be wrong if the second label fails to be extracted, etc. The whole process is severely bottlenecked by the low accuracy of the text-extraction from the OCR tool.

Code:

- MakeData.ipynb: Makes the dataset for the LSTM model. This file generates the questions and their associated types.
- Makegraphs.ipynb: makes the dataset with the chart, datatable, and associated question-answer pairs.
- ModelPredipynb: takes in the data extracted from the image and natural language question and outputs the response to the question.
- HoughOCR.ipynb: takes in the bar chart image and uses the OCR and Hough transform to extract data from the image.

Future Plans:

The overall accuracy of the code is severely hindered by the low accuracy of the Optical Character Recognition. So, in order to be able to utilize this code in the real world, I plan on looking into methods to improve the accuracy of the OCR. Potential methods may include accentuating the edges in the image to make the text more prominent. Additionally, I could find regions of text in the image, expand them and input that into the OCR. Or, maybe there is a better existing OCR available that I could use. Until the issue of extracting text is solved, this kind of application will not have high accuracy. Additionally, I plan on applying other image analysis techniques to be able to extract data from other types of figures like scatter plots, line graphs, etc.

Citations:

Bigham, J., Gleason, C., Shamma, D., Brady, E. and Guo, A., 2016. *An Uninteresting Tour Through Why Our Research Papers Aren't Accessible*.

Kahou, S., Michalski, V., Atkinson, A., Kadar, A., Trischler, A. and Bengio, Y., 2018. *FIGUREQA: AN ANNOTATED FIGURE DATASET FOR VISUAL REASONING*.

Siegel, N., Horvitz, Z., Levin, R., Divvala, S. and Farhadi, A., 2016. *Figureseer: Parsing Result-Figures In Research Papers*.

Zhou, Y. and Tan, C., 2000. *HOUGH TECHNIQUE FOR BAR CHARTS DETECTION AND RECOGNITION IN DOCUMENT IMAGES*.