Dissertation on

"VISUAL QUESTION ANSWERING ON STATISTICAL PLOTS"

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology
in
Computer Science & Engineering

UE18CS390B – Capstone Project Phase - 2

Submitted by:

Sneha Jayaraman	**PES1201802825**
Sooryanath I T	**PES1201802827**
Himanshu Jain	**PES1201802828**

Under the guidance of

Prof. Mamatha H.R
Professor
PES University

June - December 2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

'Visual Question Answering On Statistical Plots'

is a bonafide work carried out by

Sneha Jayaraman	**PES1201802825**
Sooryanath I T	**PES1201802827**
Himanshu Jain	**PES1201802828**

in partial fulfillment for the completion of seventh semester Capstone Project Phase - 2 (UE18CS390B) in the Program of Study - Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period June - December 2021. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 7th semester academic requirements in respect of project work.

Signature	Signature	Signature
Dr.Mamatha H.R	Dr. Shylaja S S	Dr. B K Keshavan
Designation	Chairperson	Dean of Faculty

External Viva

Name of the Examiners	**Signature with Date**
1. _____	_____
2. _____	_____

DECLARATION

We hereby declare that the Capstone Project Phase - 2 entitled **"VISUAL QUESTION ANSWERING ON STATISTICAL PLOTS"** has been carried out by us under the guidance of Dr.Mamatha H.R , Professor and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester June - December 2021. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

PES1201802825	**Sneha Jayaraman**	
PES1201802827	**Sooryanath I T**	
PES1201802828	**Himanshu Jain**	

ACKNOWLEDGEMENT

I would like to express my gratitude to Prof. Mamatha H.R, Department of Computer Science and Engineering, PES University, for her continuous guidance, assistance, and encouragement throughout the development of this UE18CS390B - Capstone Project Phase – 2.

I am grateful to the project coordinator, Prof. Silviya Nancy J, for organizing, managing, and helping with the entire process.

I take this opportunity to thank Dr. Shylaja S S, Chairperson, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department. I would like to thank Dr. B.K. Keshavan, Dean of Faculty, PES University for his help.

I am deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr. Suryaprasad J, Vice-Chancellor, PES University for providing me various opportunities and enlightenment every step of the way. Finally, this project could not have been completed without the continual support and encouragement I have received from my family and friends.

ABSTRACT

Question answering systems have been used in various domains and applications like dialog systems, and medical domains for interaction with patients' therapy reports, scans and X-rays. In this work ,we have encircled the domain of data analysis involving statistical plots where question answering systems can be used. The statistical charts are used on a regular basis for data visualization to interpret the data and derive meaningful inference from them. This process can be automated using question answering systems. Users can impose a question to the system, for a particular statistical chart within the scope handled, the system must then process the query along with its pointers to/from the image data and provide the answer in the most accurate manner. Building such a system requires the usage of right architecture, right frameworks/tools and huge amounts of data. This work involves the research around existing visual question answering systems for graph-plots and aims to provide alternative accounting to a novel approach. Once the model is built that satisfies the requirements, it can be deployed as a web application where a user can upload an image and, input a question, to propose the expected answer.

Visual question answering system in general, generates an answer to queries posed in natural language with the help of information extracted from the input image. Incorporating the ability to answer the questions on statistical charts is the aim of this research. In the development of our model, questions must be related to answers from fixed vocabulary or answers that can be extracted from the bounding box representation of an image or answers that can be queried from a structured table generated using visual elements. Plots related to bar plot, line plot, and dot plots and their variants have been considered to be within the scope.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Statistical charts are an intuitive and simple way to represent data. They form a way of representing structured data in the form of graphical visualizations. Such graphical visualizations aid people in better interpreting features of data. Object detection in deep learning is a field that focuses on extricating localized datum from binary or color images. Therefore, it is useful to build a model that can localize and pick up visual data in statistical plots. It is one step towards the improvement in localized object detection capabilities.

Visual plots are commonly found in research papers, scientific journals, business records e.t.c. Therefore, automation of plot analysis through the means of question-answering aids an individual to draw statistical inferences quickly from them.

The most important benefit is that visual question answering models on charts will help data analysts question and understand plots on a large scale, and automate the decision-making capabilities in several sectors such as the financial sector.

Given this motivation, the aim of the project is to build a Visual Question Answering system which accepts statistical plots along with questions on the plot with respect to the elements of the plot (such as intersection of the curves, area under the curve, median value and few other varieties of such relational queries) and provides answers to the questions posed.

The system should discover relationships between elements of a plot and provide relational reasoning to answer questions on the plot.

Given an image of a statistical plot and a corresponding question, the model must be able to generate a representation of the image, parse it into an intermediary that is well interfaced with the workline , understand the query, and generate a suitable reply.

Therefore, it involves an understanding of localized image-element and the query language to be able to provide for visual reasoning. This work however restricts its scope to a certain number of selective plots and their inner variants that are frequently occurring in most common data representations. Plots related to bar plot, line plot, and dot plots and their variants have been considered to be within the scope.

CHAPTER 2

PROBLEM STATEMENT

Statistical plots are used widely by academicians and business employees because they are a simple way to represent data. They can be easily analyzed and interpreted.

What if we could build an automated system that can analyze, discover relationships between elements of a plot and provide for relational reasoning capabilities or simply answer the queries posed on them? Such a system would mark a step towards machine reasoning capabilities.

With this motivation, the project aims to build a Visual Question Answering system that accepts statistical plots along with plot-specific questions concerning the elements of the underlying plot, such as the data-retrieval, mean, median, range, min-max, difference, comparisons and few other varieties of such relational queries, to provide answers.

The difficulty with statistical plots is that even though they are images, they contain both structured and unstructured data. In the case of natural images, there are just visual elements to handle. However, that is not the case with statistical plots, since they contain both visual elements in the form of bars/sectors and textual elements in the form of axis labels and ticks. In short they contain multiple objects within an image where the object is a plot element and the image is the graph itself. To add to this, the size of objects that are there in natural images are constrained to small, medium and large in general. In the case of statistical plots, the aspect ratio is much more varied. For example, in the case of bar plots, there could be bars that are extremely small, and bars that are extremely large on the same plot.

The measure of the accuracy of prediction in the case of images is normally IOU (intersection over union). The success criteria for a correct prediction in the case of natural images is normally 50 per cent. The same rate is insufficient for statistical plots. This is because we want the prediction for a bar value (in the case of bar plots) to correspond to the actual value as seen on the graph, as close as possible. Therefore the adequacy criteria for a successful

prediction is much higher.

The above-mentioned factors highlight the differences in natural images and statistical plots. Therefore, state-of-the-art object detection models do not suffice for this application. The aim here is to apply deep learning concepts to come to an acceptable solution to the problem of analysing statistical plots using machines. In this work we make use of object detection algorithms rather than image detection algorithms such as faster R-CNNs with feature extractors which are localized, FPN, deep nets like Resnet 101, 50 or Resnext to constitute an object detection model followed by a set of utility models to perform character recognition concluding with a question answering module.

CHAPTER 3

LITERATURE SURVEY

In the following subdivision, we present the current understanding and knowledge of the area along with reviewing substantial findings that help shape, inform and reform our study leading us to set the platform to further envisage the possible improvements that can be brought up.

3.1 Background on Statistical Charts modeling for QA system

This section briefs the papers consulted and thoroughly reviewed to gain information on background, data used, the architecture style, the patterns and the proposed/existing methodologies being used in the domain of question answering systems for statistical charts.

3.1.1 Answering Questions about Charts and Generating Visual Explanations [1]

Summary

The paper under consideration proposes a chart question answering system that generates chart specific answers along with the explanation on how the answer was obtained. The visual attributes of the charts are transformed into references to the data. State-of-the-art ML algorithms are used to generate answers and their corresponding explanation.

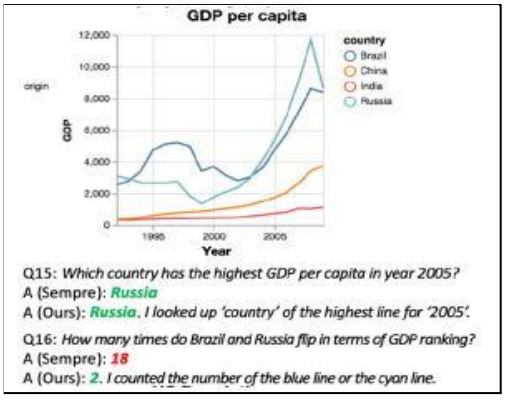


Figure 3.1

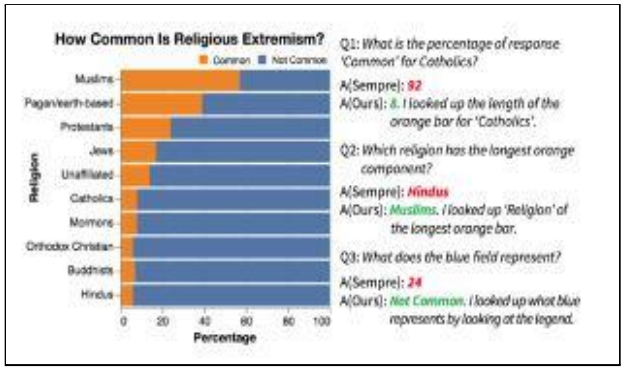


Figure 3.2

Figure 3.1 illustrates two question and answer pairs for a line plot. The results are compared between the Sempre model and the model proposed.

Figure 3.2 illustrates three sample question and answer pairs for a stacked horizontal bar plot.

Dataset

The compilation of the dataset consists of 52 charts, congregated from four contrasting sources:

- The Vega-Lite Example Gallery
- Graphical Charts in Pew Research Reports
- D3 charts that are accumulated from the internet
- Charts fabricated from tables present in the WikiTableQuestions data compilation.

The questions, answers and their explanations were manually generated.

Dataset Counts :

In union, the compiled data includes 5 line charts and 47 bar charts (32 simple, 8 grouped, 7 stacked). In total, 52 charts are synthesized that include 629 questions, 866 answers and 748 explanations for the answers generated. This data is considered for our work as well.

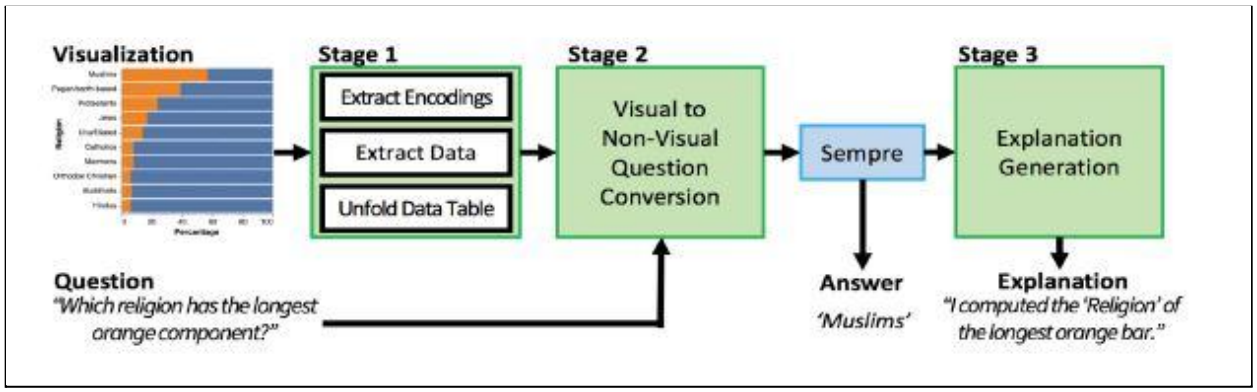


Figure 3.3

Religion	Response	Percentage
Muslims	Common	57
Muslims	Not common	43
Pagan/earth-based	Common	39
Pagan/earth-based	Not Common	61
⋮	⋮	⋮
Hindus	Common	6
Hindus	Not Common	94

(a) Flat data table

Religion	Common	Not common
Muslims	57	43
Pagan/earth-based	39	61
Protestants	24	76
Jews	17	83
⋮	⋮	⋮
Buddhists	7	93
Hindus	6	94

(b) Unfolded data table

Figure 3.4

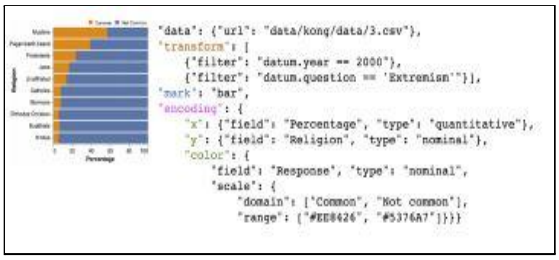


Figure 3.5

Figure 3.3 illustrates the pipeline of this model for the question answering system, ***Figure 3.4*** shows the dataset format and ***Figure 3.5*** shows the unfolded data table.

Methodology Proposed

Firstly, visual encodings like the pinnacle aka height of the bar, color/shade grading of the line, etc. are extracted from the charts. The input question is transformed, replacing any visual references made by chart elements to the non-visual references to the data fields and data values. The Unfolded table is passed through Sempre (QA algorithm that functions with relational data tables instead of any statistical charts) to generate the answer. The Sempre model converts the input question into a lambda expression. It then performs query execution on the data table generated to produce the answer. The lambda expression obtained is transformed into a visual explanation for the answers by a method known as template-based translation.

Formal Steps Stated in the paper

- **Stage 1**: Extract Data Table and Encodings
- **Stage 2**: Visual to Non-Visual Question Conversion
    - Step 1: Mark detection

- Step 2: Dependency parsing
- Step 3: Visual attribute detection
- Step 4: Visual operation detection
- Step 5: Apply encodings
- Step 6: Natural language conversion

● **Stage 3**: Explanation Generation

- Step 1: Natural language conversion
- Step 2: Implicit field recovery
- Step 3: Redundancy Cleanup
- Step 4: Sentence Completion
- Step 5: Encoding application

Merits

The paper not only provides accurate answers to the questions but also provides an explanation on how the answer was obtained. The model produces valid answers and their corresponding explanations as opposed to the Sempre model that could not answer the questions correctly.

Demerits

The system cannot handle certain types of questions that involve synonyms of the features present in the chart. There is scope for improving the explanation provided for the answers.

3.1.2 FigureNet: A Deep Learning model for Question Answering On Scientific Plots [2]

Summary

This model uses a CNN with depth-wise convolutions, LSTM and feed-forward NN to handle the task of answering questions on plot such as pie and bar on a dataset that's named FigureQA.

Dataset

The dataset used is the FigureQA dataset that contains more than a million questions with answers on various types of scientific plots. This dataset has plots with elements that are color-coded. There are a total of 100 colors that are used across both training and test datasets. Therefore, it is possible to distinguish between elements without the need of character recognition for text. Additionally, it provides pre-annotated data with bounding boxes.

Template
Is X the minimum?
Is X the maximum?
Is X the low median?
Is X the high median?
Is X less than Y?
Is X greater than Y?

Figure 3.6

Figure 3.6 shows the template of questions in the FigureQA dataset.

Model

The FigureNet architecture, as proposed, can handle the task of answering relational questions on pie charts and bar plots. It uses the FigureQA dataset, which consists of statistical plots with plot elements that are color coded. Additionally, it is guaranteed that the plot consists of no more than 11 plot elements, and that there are 100 different colors that are used to represent plot elements. The end goal of the FigureNet model is to be able to answer questions in a binary yes/no manner. To be able to do this, the authors have divided the task into subtasks as follows.

- Spectral Segregator Module - Identify plot elements and color of the plot elements.
- Order Extraction Module - Identify and quantify the values associated with each plot element, and then sort it into increasing order.
- Question Encoding - Provide an encoding for the question.
- Question Color encoding - Identify mentions of color in the question.

Methodology proposed

Spectral Segregator Module:

This module is used to identify individual elements and the color of these elements of the plot. 128 x 128 x 3 image is passed as input to a CNN that uses depth-wise convolutions to identify colors and separate channel information. This way we don't just get an aggregate map of the image. The output here is a 512 dimensional image representation. This image representation is passed to a 2-layer LSTM to get the most probable color for each element. There can be at most 11 elements in a plot.

Order Extraction Module:

This module is used to identify and quantify the statistical values of each plot element and their relative order. It is similar to that of the previous module except that now the output of the LSTM will be the ordering for each of the plot elements starting from 1.

Question Encoding and Question Color encoding:

This module uses 2 layers of LSTM cells (many to one model) to produce a question encoding.

Final feed-forward NN:

All the four modules are concatenated and passed onto a feed forward NN to produce a binary (Yes/No) answer using the Sigmoid Activation function for the output layer.

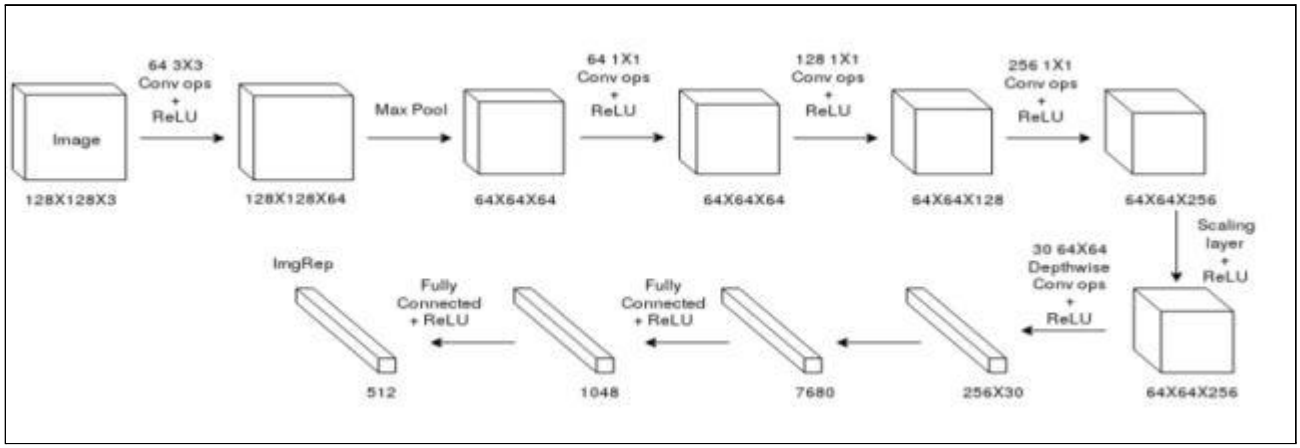


Figure 3.7

```
[Royal Blue, Aqua, Midnight Blue, Purple, Tomato,
STOP, STOP, STOP,
STOP, STOP, STOP]
```

Figure 3.9

Figure 3.7 shows the architecture of the Spectral Segregator Module that uses layers of

convolution and max pool, followed by depthwise convolutions and feed-forward layers.

Figure 3.8 shows the rest of the spectral segrator module that uses a custom LSTM architecture. The input here is the image representation that is obtained as the output from the architecture in Figure 3.7.

Figure 3.9 shows a sample output as obtained from the architecture in Figure 3.8.

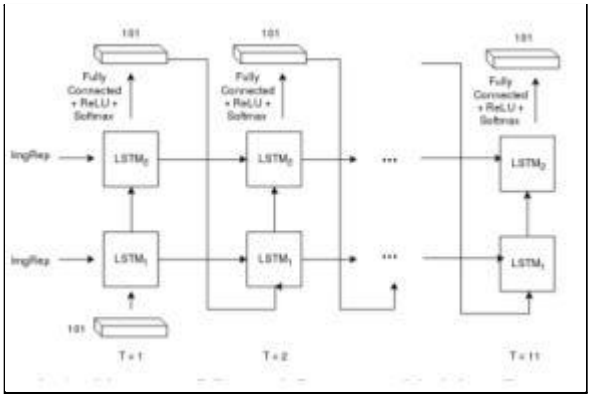


Figure 3.8

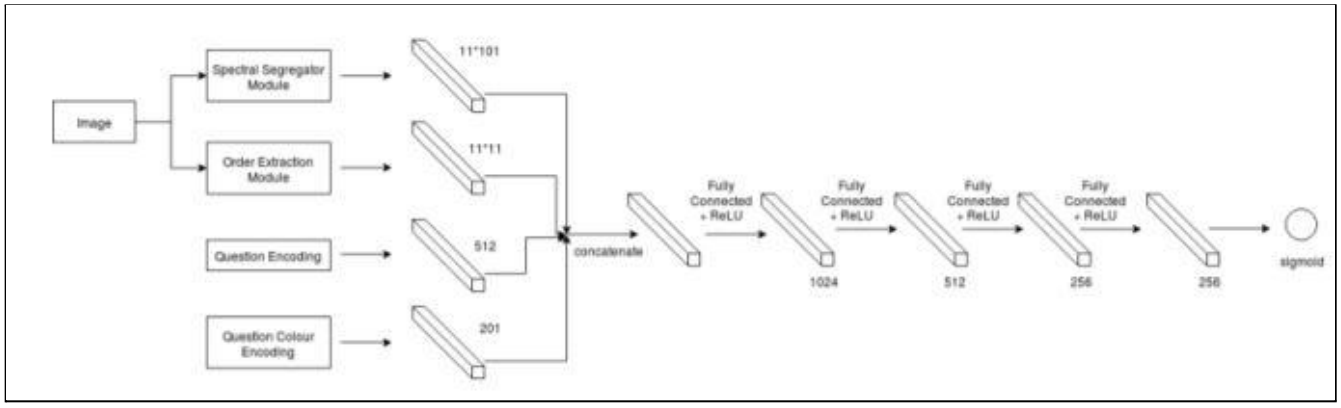


Figure 3.10

Figure 3.10 shows the final feed forward architecture.

Figure Type	*CNN + LSTM*	*RN(Baseline)*	*Our Model*	*Human*
Vertical Bar	60.84	77.53	**87.09**	95.90
Horizontal Bar	61.06	75.76	**82.19**	96.03
Pie Chart	57.91	78.71	**83.69**	88.26

Figure 3.11

Figure 3.11 depicts a table comparing accuracy values for each type of plot.

Merits

The model performs significantly better than the baseline models. This is because the architecture doesn't use the traditional CNN, instead uses depthwise convolutions. Additionally, the model used less training time as articulated in the paper.

Demerits

The model works on only bar plots and pie charts. It is capable of only binary reasoning, and not capable of answering open-ended questions. It makes use of the FigureQA dataset, thereby making use of the property of the charts being color-coded.

3.1.3 ChartNet: Visual Reasoning over Statistical Charts using MAC-Networks [3]

Summary

Proposed paper solves the problem of reasoning over charts (only bar and pie charts) using MAC-Network (Memory, Attention, and Composition). The model is capable of answering open-ended questions and gives chart-specific answers. The classification layer of MAC is substituted by the regression layer and constructs a boundary for the text that corresponds to the answer. OCR is used to read the text and display the answer.

Dataset

The data synthesized by the model consists of bar-charts and pie-charts. The bar charts dataset consists of vertical bars and was created by varying the height and number of colors of bars. Data for pie-charts is created by varying the colors of sectors and also the angles between them. The annotations of the bounding box that are present over the chart images are saved to give chart-specific answers. In total, 20k, 5k and 5k image question pairs for training, validation and testing are created, for both bar charts and pie charts.

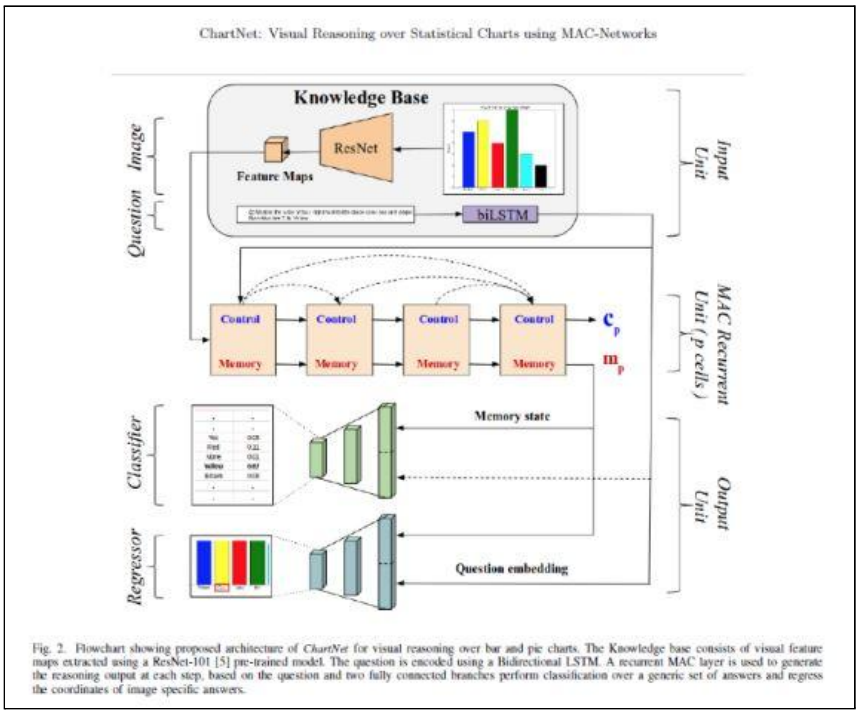


Figure 3.12

Figure 3.12 shows the architecture of the model proposed.

Methodology proposed

ChartNet network consists of three layers: Input unit, MAC Cell, Output unit.

Input Unit

Bar or pie chart is given as an input and corresponding question. Features from the images are extracted using ResNet101 deep CNN architecture. Knowledge base is defined to depict the height and the width image. Questions are converted into word embeddings and they are further processed using the biLSTM model.

MAC Cell

It represents a recurrent unit which consists of three components: Control, Read and Write. It is defined to reason the questions posed and also to implement them.

Output Unit

This unit consists of two networks: Classifier and Regressor. Classifier network predicts the probability distribution over all of the predefined answers by using softmax normalization. The regressor network is used to provide chart-specific answers.

Merits

Automated method for question answering over open-ended questions. The MAC-Network included with the regression layer helps the model make predictions over unseen answers.

Demerits

The model is not generic and works only for vertical bar charts and pie charts. Model cannot answer questions that require numerical operations.

3.1.4 PlotQA: Reasoning over Scientific Plots [4]

Summary

A step towards developing a holistic plot based visual question answering model, which can handle both in-vocabulary and open ended queries using a hybrid approach.

Dataset

The graphical summaries are produced from data provenanced from organizations like the World Bank, government maintained sites to name a few, thereby having a large vocabulary of graph parameters like ticks, and a wide variety of range in data instances. Out-of-vocabulary questions are generated and they are not straightforward as they are generated on the basis of 70 plus patterns extracted from 7,000 public flock questions asked by data collectors on a sampled set of 1000+ plots.

Datasets	#Plot types	#Plot images	#QA pairs	Vocabulary	Avg. question length	#Templates	#Unique answers	Open vocab.
PlotQA	3	224,377	28,952,641	Real-world axes variables and floating point numbers	43.54	74 (with paraphrasing)	5,701,618	Present

Figure 3.13

Answer (A) Type	Question (Q) Type		
	Structure	Data Retrieval	Reasoning
Yes/No	36.99%	5.19%	2.05%
Fixed vocabulary	63.01%	18.52%	15.92%
Open vocabulary	0.00%	76.29%	82.03%

Figure 3.14

Figure 3.13 summarizes the dataset used. ***Figure 3.14*** shows the long range distribution of Query and Response types from the data compilation in the PlotQA data compilation.

Model

Existing Works :

Existing solutions for VQA fall under two categories: (i) extricate the response from the graphical data input (like in LoRRA) or (ii) respond with an answer to the query posed based on the existing vocabulary (like in SAN and BAN). Such approaches seem to work well for data compilations as portrayed in DVQA, but under fit for PlotQA with a considerable majority of out of vocabulary queries.

PlotQA's Model :

This is a composite model encompassing the below features and entities:

(i) a binary categorizer for decision making as to whether the input query be responded with from an existing vocabulary or demands an advanced treatment.

(ii) a simpler question categorizer to respond to queries of the simpler or complex treatment type.

(iii) The process pipeline with the CNN and LSTM combination setup.

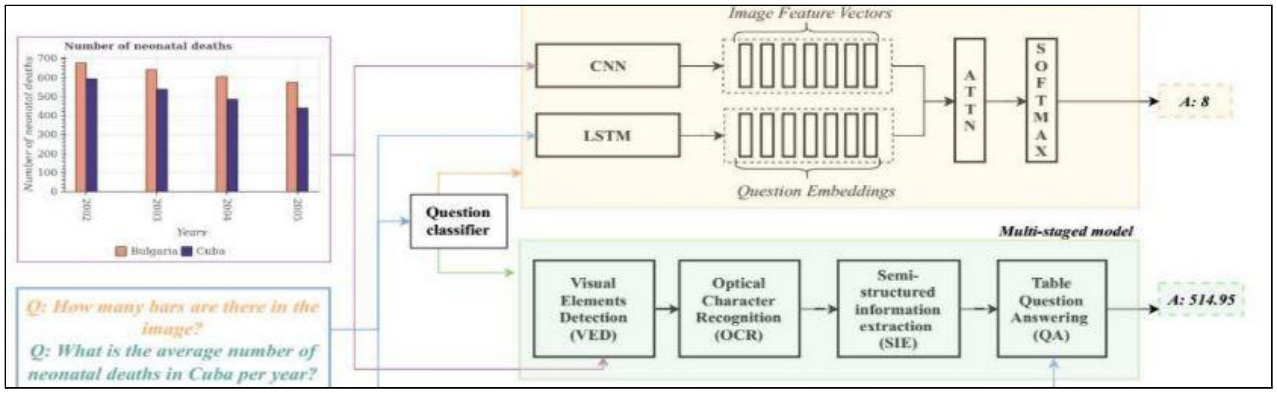


Figure 3.15

Figure 3.15 shows the architecture of the model proposed. Observe the two pipelines.

Methodology proposed

This is a composite mix match model encompassing the detailed entities as follows: (i) a binary classifier for categorizing the complexity of the question and to provide decision about if it can be handled by in vocabulary or does it need assistance of out of vocabulary processing pipeline (ii) a simpler question categorizer model to respond to queries of types as mentioned in (i), finally (iii) presence of a multi-staged model encompassing four software modules as briefed in the following section to deal with the lower half of the pipeline which is the out of vocabulary questions.

1. Visual Elements Detection Module

Primary task to perform is to extricate the visual entities by demarcating / annotating bounding boxes around those entities and categorizing them into the appropriate groups.

Upon comparing all methods, it is observed clearly that the Faster R-CNN model in union/combination with Feature Pyramid Network (FPN) outperforms the existing architecture combinations and hence becomes an apt fit for the visual element detection module.

2. Object Character Recognition Module:

The common visual entities of the graphical summaries like legends, tick labels to name a few, accommodate numerical and textual data. For the purpose of extracting this data from bounding boxes annotations, the avant-garde OCR model is used.

3. Semi-Structured Information Extraction Module:

The penultimate phase of the pipeline. The data captured as results in the form of json/dictionary from the previous phase is formatted into a table structure using this module.

4. Table Question Answering Phase:

The ultimate final phase of the processing pipeline is to answer queries by superimposing them on the semi-structured pivoted table version of the image which has now been dissected. This is akin to answering questions from the WikiTableQuestions dataset, so the similar process is recreated to obtain results.

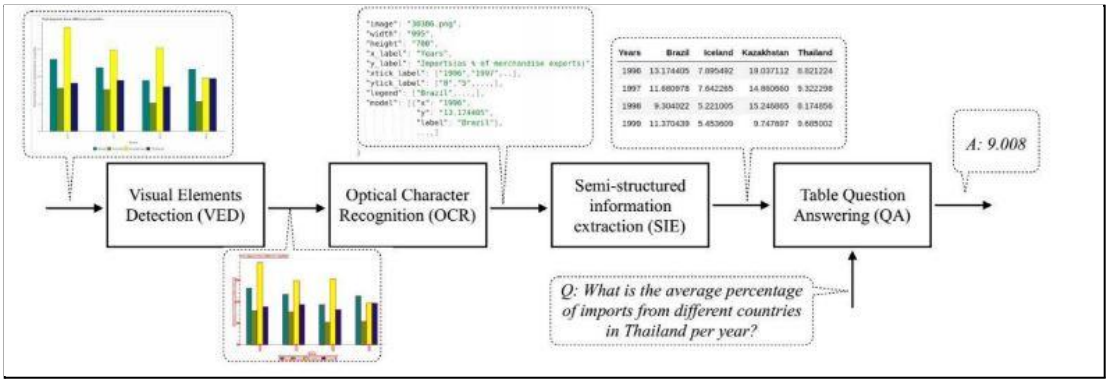


Figure 3.16

Figure 3.16 shows the proposed multi-staged modular/unit-wise pipeline.

Merits

Can handle out of the vocabulary questions (OOV) along with in-vocabulary question types. The data collected to prepare graphs in the dataset are from various financial and business resources. Blurs the line of difference between computerized-data plot datasets and real life data summarized in graphs and query patterns.

Demerits

The model is not generic and works only for bar charts , line charts and dot plots. There exists a need/development in regard to more precise visual element detecting (VED) modules to enhance responses over the queries posed on the plots.

CHAPTER 4

DATA

The following chapter describes the data used to build the question answering system for statistical plots. Building the right questionnaire with right visualization is critical to build a high accuracy model.

4.1 Overview

Statistical plots are used to represent the data and help in deriving insights from them. Some of the basic charts are bar charts, dot plots and line charts. Visual aid is required to analyze these charts and extricate the objects/plot members from them and answer some of the questions related to them.

4.2 Data Format

Statistical plots are used to learn about data and their important features. Building a visual question answering system requires statistical plots and well-defined questions and answers. The detailed description of the charts and the question answer pair is provided in further sections. The over data compilation comprises graph images which are supported by a set of annotation files that are well formatted using a json structure. The images and their annotations must be correlated/combined to get the bounding boxes around the training images, that is the preparation step before the (image + annotated) data is passed to training for object detection.

4.3 Statistical Charts

Statistical charts are one of the inputs to the visual question answering system. The types of charts that we have constrained to are bar charts, line plots and dot/dot line plots.

A typical plot in general irrespective of its type will include

- Title of the plot/graph
- Xlabel and Ylabel
- Legend data if any
- Xticks and Yticks

There are different types/variants of bar charts like vertical charts, horizontal charts, simple, and group charts. Whereas dot plots and line plots have zero or no variants.

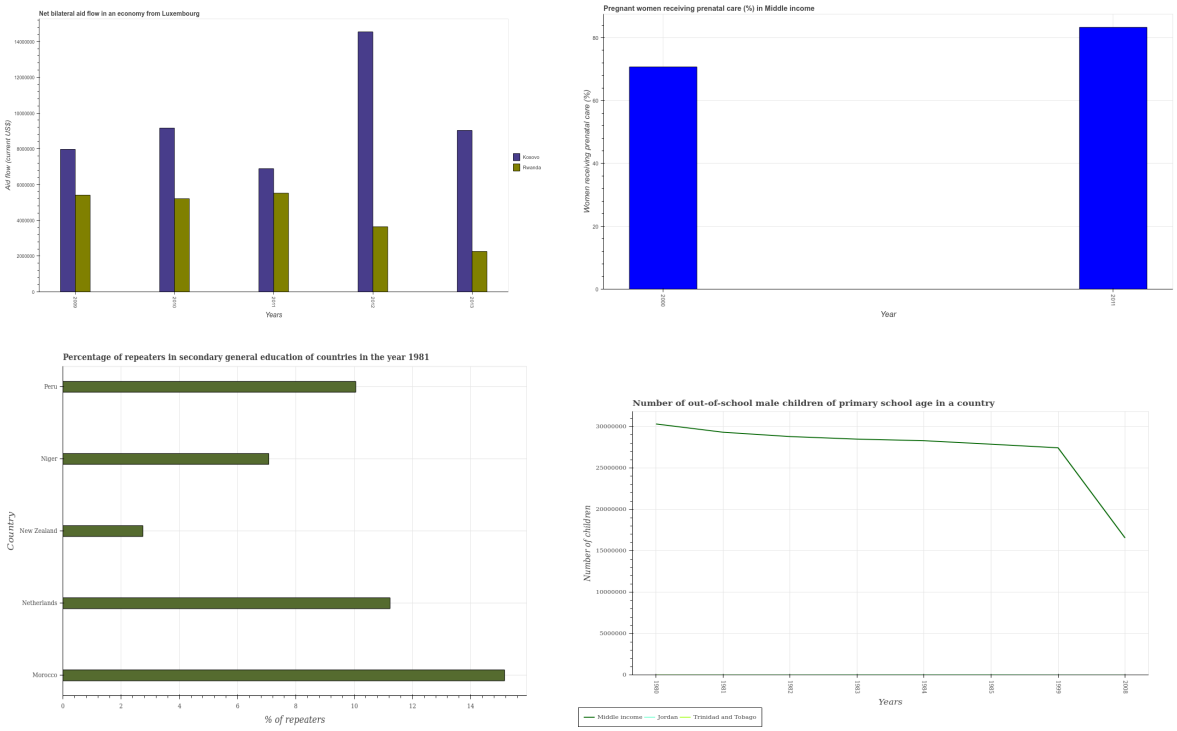


Figure 4.1 (Image Courtesy:PlotQA: Reasoning over Scientific Plots)

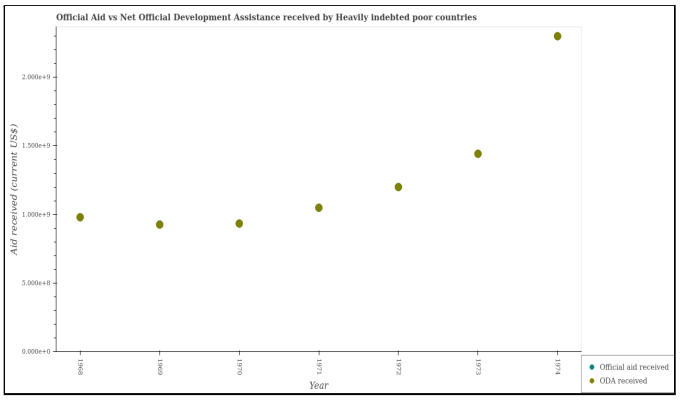


Figure 4.2 (Image Courtesy:PlotQA: Reasoning over Scientific Plots)

Figure 4.1 represents [top-left to bottom-right] vertical grouped bar , simple vertical bar , simple horizontal bar and simple line plot. ***Figure 4.2*** represents a dotplot.

The plot element detection stage will intake an excess of these images with their corresponding annotations, the annotations of the images act as a catalog for each image, they have information regarding the bounding box points of reference for every significant plot element present in a graphical image. With the help of this data, our object detection model tends to learn the possible bounding boxes location within any inference image passed on to it.

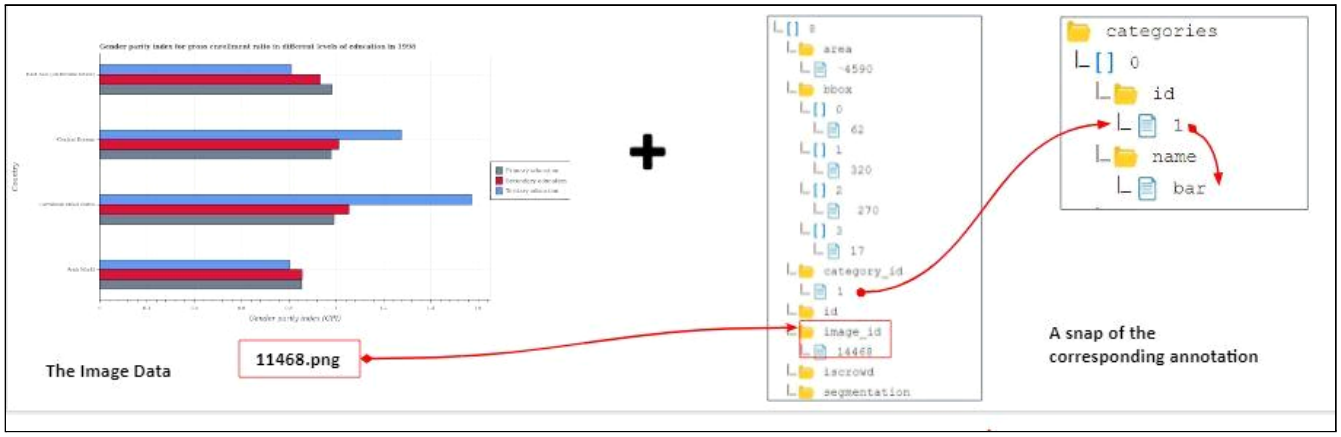


Figure 4.3

Figure 4.3 shows the raw image data and the corresponding annotation data which acts as a catalog for the image. It is to be noted that both of them exist separately. Necessary preprocessing is done to combine them together, so that the plot elements have their bounding box point around them before training.

4.4 Question And Answers

Every image has a list of questions and their answers stored in the repository. Descriptive questions or Open-ended questions are generated for each image. The answers for them are also stored. Some of the examples of the question answers are shown in the figure. During the training of the model, the chart list of questions is passed as the input. However, the question input is only used in the final stage wherein it is fed into a weak supervised table question answering model. The model for table question answering is already pre-trained in case of this work to predict the answer. During testing, only the chart and a question is taken as the input, and the model predicts the answer.

At a lower level, firstly the image is passed onto a trained model and inference is performed to get the bounding box information of all the involved plot elements, next the points of references to the graph elements is passed along with the image to the OCR stage , in which the localized text content is extricated, after which we have utility modules to map the complete image data into its tabular equivalent and generate the csv. Now things become simpler because the plot data is no more an image but it is a semi-structured table on which table question answering can be performed, just as in the case of the wikitables questions dataset. The questions that are handled in our work are related to data-retrieval following a simple select, project or select cum project vice-versa, statistical mean, min - max, boolean truths, range based queries, simple summations, summations across an interval from the graph data, trend wise queries, differences with few constraints and restrictions laid on them.

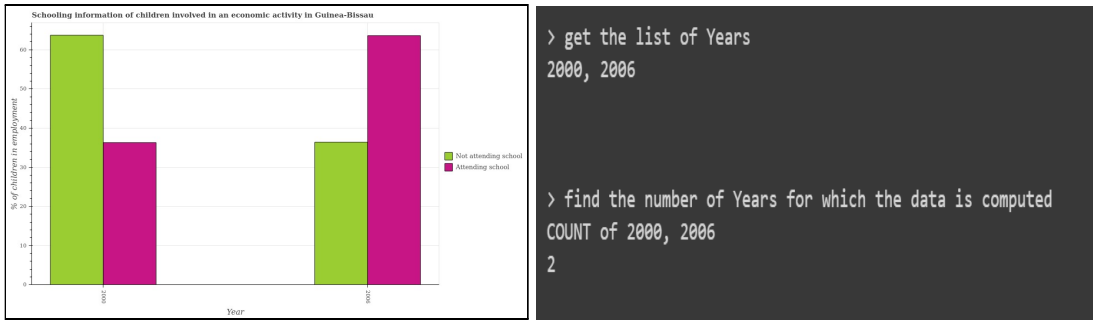


Figure 4.4 indicates a graph {grouped vertical bar} on the left and the question posed on it

with the predicted answer on the right.

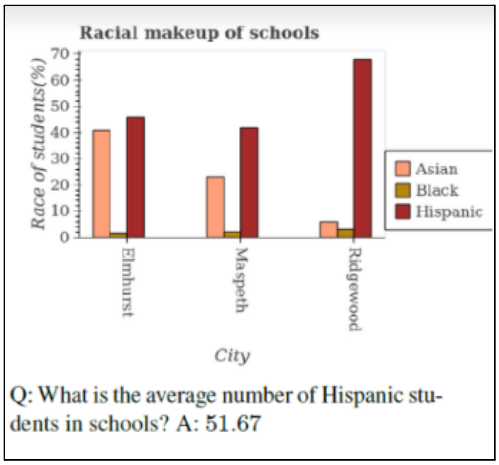


Figure 4.5

Figure 4.5 is a sample of a query associated with the graph concerning a numerical/arithmetic operation of averaging a particular category.

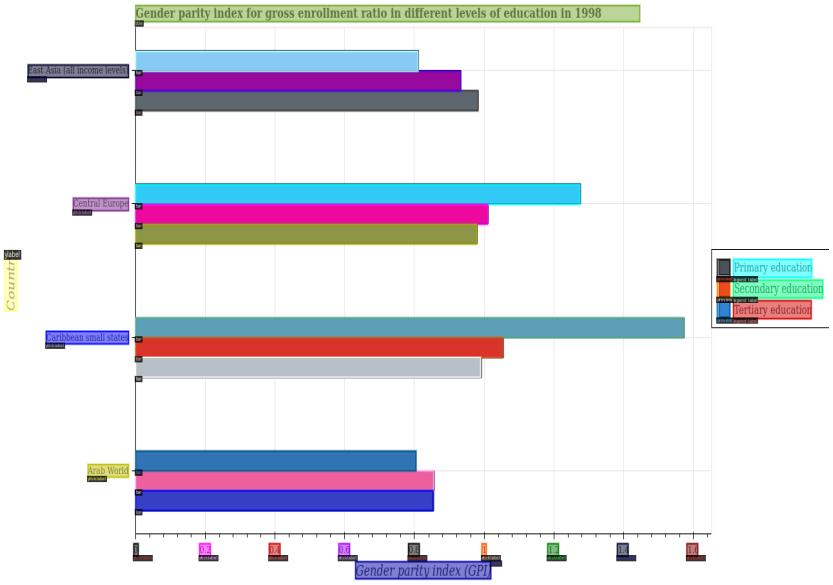


Figure 4.6 The output of an image in Figure 4.3 superimposed with its annotation (pre-training) . Those are the bounding boxes which were generated manually

CHAPTER 5

SYSTEM REQUIREMENTS SPECIFICATION

5.1 Project Scope

Goal

The system should be able to answer questions on Vertical and Horizontal Bar Graphs {simple, grouped}, Dot-Line Charts, Dot plots and Line Plots and questions within the scope of handling. Provide an user interface through which the end user can upload the image and a corresponding question.

Limitations

The system will not be able to answer questions on the smoothness or the roughness of the plots. It can only answer relational queries – queries with respect to the other elements of the plot. Handles only graphs considered within the scope of work and there do exist some constraints on the questions posed on the graphs. The questions that are handled in our work are related to data-retrieval following a simple select, project or select cum project vice-versa, statistical mean, min-max, boolean truths, range based queries, simple summations, summations across an interval from the graph data, trend wise queries which have certain keywords required to be present in the queries.

5.2. Product Perspective

Visual question answering specific to statistical plots has less prevalent work done. It is one step towards the improvement of machine reasoning and pattern identifying capabilities and object detections in scientific plots.

5.2.1 Product Features

1. Input: The input to our model is an image of a statistical plot covered within the scope of work and a corresponding question on the plot.

2. Model: Our model will take in the input, process the visual image using the object detection model further converting the objects obtained into a consolidated semi-structured format and parse the query, concatenate the inferences from both and produce an output. Thus, it combines the technicalities of object detection capabilities and query language understanding.

3. Output: The output of the model is the answer to the question.

5.2.2 Operating Environment

The model will be made available as a web application; hence it does not depend on the underlying Operating system and its versions. It only requires browser support of HTML5 and above with a flask oriented backend to serve the requests consisting of input image and supporting query. Use of the model can be done via the internet. Colab Pro offers a Linux environment, however the OS utilities are used only for the backend functionality, none of that affects the user interface.

On the server side, the model can be saved and loaded when necessary (post training, the weights are saved to perform inference and generate further outputs for intermediate stages). Therefore, the platform must be available during requests , must provide sufficient disk space of minimum 15GB for hassle free loading of dataset and saving intermediate results, RAM of minimum 12GB for smooth processing without hiccups, GPU support of 16GB to get the deep nets trained over a period of time for longer iterations. For the training and testing phase a linux based environment is needed and it has been done so. A linux backend plays a major role in serving the request, but the technicalities are hidden and unrelated to the end user.

5.2.3 General Constraints, Assumptions and Dependencies

The project focuses on model building and providing for accurate and reliable answers to the questions posed on the statistical charts. Hence, there is less focus on the security considerations. However, our solution will consist of an interface to the model that facilitates image upload and questions on the image. Most of the assumptions, limitations and dependencies are covered in the section regarding the limitations. The system produces fair results only on the graphs and queries handled well within the scope.

5.2.4 Risks

Operational risk in terms of management and support for the product is a possible risk case.

Functional Requirements

Question Answering System for Charts take in statistical charts and questions related to them as input. Visual features from the charts have to be extracted and preprocessed. This can be done using techniques like object detection processing and Optical Character Recognition (OCR). The questions provided as an input need to be of the type handled by the scope of this model. Important details from the image and the questions need to be mapped (done here through OCR and semi-structured table creation) and the corresponding answer should be predicted (table question answering). Deep learning methods and architectures will be employed to predict the output both at the image level and questionnaire levels. Inputs are validated by the system to recognize only statistical charts. Any other images will be responded with poor results as they aren't seen by the model previously or rather untrained. The system will be trained on huge amounts of data and the results will be validated against the true answers using suitable methods. The parameters for the object detection model will be tuned to provide the most bounding box capturing capability and a proper pre-trained table question answering module will have to execute the queries on the tabular data and respond with an answer to the end user.

5.3. External Interface Requirements

5.3.1 User Facing Interfaces (UI)

The user interface is a web application which will take a chart and user-specific questions. Users will be allowed to upload an image from their local drive. A text box will be provided that will accept the questions. The model will take the images and questions as the input and run them in the backend, where the low level operations will be performed to feed in the image and questions to the pipeline. It will return the most probable answer and display it on the screen. Additional data like the confidence of the answer can also be displayed. A trained model will be deployed on the web server, and hence, the output should be produced within a few seconds.

5.3.2 Hardware Requirements

Deep learning tasks are computer intensive. The hardware requirements to build and train the model will require a minimum of 12GB RAM. A disk space in excess of 10GB for hassle free operation and a GPU support in excess of 10GB. Optionally, the model can be trained on the cloud to avoid physical hardware limitations, in platforms that offer compute as a service. Once the model is trained, the testing phase can be done on the same cloud commodity using the pay as you go method.

5.3.3 Software Requirements

Question Answering Systems are built using deep learning models and NLP techniques. Python (Version: 3.6, 2.x or more), NLP libraries, object detection toolkits such as detectron, open source deep net frameworks, image processing tools and basic os and python-ic utilities supported with intermediate helper modules. Web frameworks like React or Flask are required to build and deploy the web application. The system can be built on an OS with linux distribution. Versioning of the model will be done using GitHub or possibly a dedicated Google account can

be provisioned to carry out the activities, because there is a high amount of storage required to maintain a repository. Separate notebooks can be created as per requirement with helpful bookmarks to make it hassle free for the users to run through them.

5.4. Non-Functional Requirements

5.4.1 Performance Requirement

● **Usability** : The trained model must be available at ease to use it, just by choosing an image input from the local drive or file-system. Similar to drag and drop or attaching files. The user must be able to navigate through the interface even with minimal exposure towards computing technologies.

● **Reliability** : As the product is developed by following practices in deep-learning, machine learning and imaging, there is no certain fixed level of reliability that can be set for the product. Reliability is not completely independent of the inputs to the model, hence reliability varies with respect to the context and type of inputs passed in, provided they comply with the assumptions and restrictions of the model.

● **Maintainability** : As the product is just a model, maintaining the model isn't a difficult task, it only requires a cloud machine, a drive to support storage on cloud so that it can persist and to reside on and a browser/interface to access.

● **Performance** : The model is expected to draw statistical inferences based on the question and the input passed with a good and acceptable accuracy level relative to ground truth or human evaluation.

● **Robustness** : The model must be robust enough to handle any of the graph images and questions related to its scope of operation. The model must yield good performance throughout

the pipeline for any input from the wide range of possible inputs pertaining to the scope of the model.

5.4.2 Safety Requirements

Safety requirements pertaining to this product are minimal as it isn't deployed onto a live physical environment. The decision process for users, such as passing image inputs or attaching them to gain access to model/product and data must follow a need-to-know principle, which states that access to internal data must be available only to the designers of the model and shouldn't be exposed to the end users.

5.4.3 Security Requirements

Data Sharing : As there are lots of graphical images to be collected, the data and statistics storage along with logger data will be done to maintain the correct functioning of the model and to reconstruct what went wrong in case of any system failures by constructing checkpoints. The datasets, if artificially generated, need to be secured locally and not be made available for commercial usages.

Model security : The model trained needs to be protected on a local machine or on cloud storage if deployed onto the cloud. Modification of ownership to only allow viewable access can be enabled to outsiders; versioning of the model weights can also be done to build over the training and enhance the performance.

CHAPTER 6

DETAILED SYSTEM DESIGN

6.1 High Level Design:

Question Answering System for Charts take in statistical charts and questions related to them as input. Visual features from the charts have to be extracted and preprocessed. This can be done using techniques like image processing and Optical Character Recognition (OCR). The questions provided as an input need to be pre-processed using NLP techniques. Important details from the image and the questions need to be mapped and the corresponding answer should be predicted. Deep learning methods and architectures will be employed to predict the output. Inputs are validated by the system to recognize only statistical charts. Any other images will be rejected by the system or rather the model doesn't recognize the graph image. The system will be trained on huge amounts of data and the results will be validated against the true answers. The parameters for the model will be tuned to provide the most optimal answer as the output.

6.1.1 Current existing works

There have been attempts in the recent past to improve machine reasoning capabilities through visual question answering systems on graphical plots. The RNN architecture and the CNN-LSTM architecture form baseline comparison models with an accuracy of 75% and 60% respectively. This in comparison to human accuracy falls short by a large margin. A recent paper publication introduced the FigureNet architecture that was able to achieve an accuracy of approximately 85% on an open-source dataset. This however, only gives a yes/no binary output to a question posed, and is limited to only bar and pie charts. Another adaptation to this model showed significant enhancements in terms of being able to answer open-ended questions on a different synthesized dataset. This domain of visual question answering on statistical plots, however, has a lot of scope of improvement in terms of future enhancements of these models. There is a possibility of expanding the types of charts to those beyond bar and pie charts or even improving on accuracy through model adaptation. In our work we made an attempt to finetune the existing work and also tried out the alternatives for the table question answering stage.

6.1.2 High Level System Design

High Level Design Diagram:

Input: There are two inputs to our model that the user needs to provide. The first input is an image – that depicts a statistical plot and the second input is a relational question on the image.

Our Model: The design consists of **3 primary components**.

1. Image Encoding Module / Plot Elements Detection model

This is a module that takes in the image as its input along with the annotations of the image, correlates them together. This module consists of a deep learning FASTER R-CNN module for object localization, because our main aim is to localize and produce bounding boxes around the plot elements and extricate them rather than classifying an image. This module produces the bounding box annotation of all involved plot elements captured by the object detection model which is fine-tuned and trained by us. Bounding boxes are a vector representation of the plot elements in a graph, which refer to the coordinates of the object (top_x , top_y , bottom_x , bottom_y); we effectively need only these 4 points to draw a bounding box around a plot element. Bounding box values of the plot elements like xlabel, title, ylable, the bar, the line, the dot are all obtained. While an image classification network can tell whether an image contains a certain object or not, it won't say where in the image the object is located. We are in the quest of **locating the plot elements** and **not just classifying the plot.** Hence we don't need just a CNN, we need an object detection neural network like R-CNNs we need to decide over the **object detection model**.

- We chose **Faster R-CNN** as our object detection model which is a **descendant from the R-CNNs series**.
- The heuristic behind doing so can be inferred from the observations below and also due to the presence of the RPN (Regional Proposal Network is known for locating feature targets accurately).

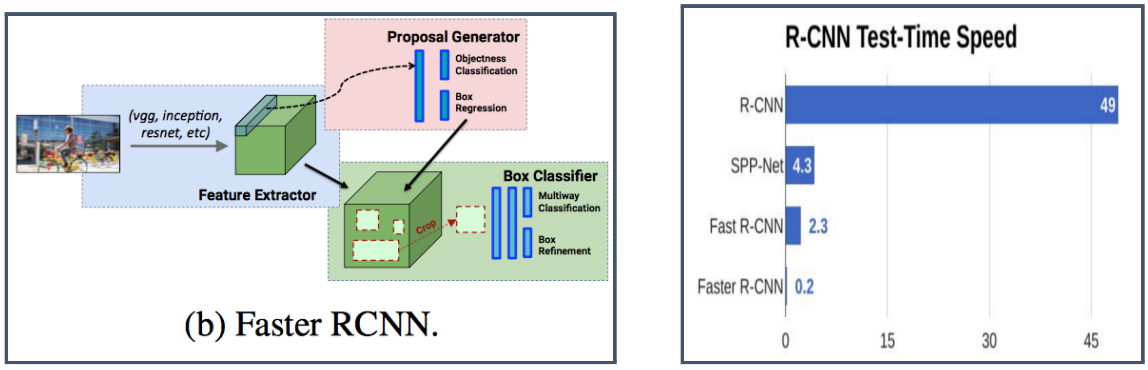


Figure 6.1 Indicating the Object detection model we chose

We also need to decide on the feature extractor model for serving as the backbone/feature-extractor for our faster-RCNN Setup. We Chose **Resnet-101** to be our feature extractor due to skip connections and residual blocks which can reduce the problem of vanishing gradients in a very deep network like Resnet-101.

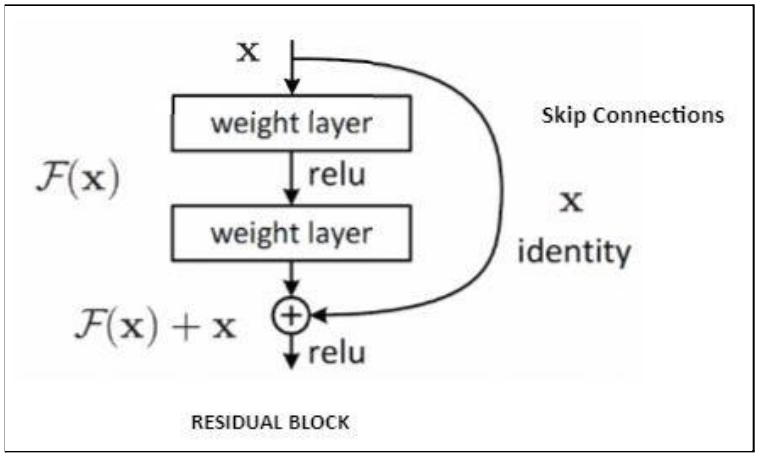


Figure 6.2 Indicating skip connections in a Resnet-101 model

2. Question Encoding Module / Table Question Answering stage

The input to this module is the question (English Language) and the output is a question embedding. The question embedding captures all of the relevant information in the question in a format that is suitable for further modeling. This module needs tabular data on which the question answering can be performed. Hence the output of the image module must be further

structured using other utility stages so that a semi structured table is obtained, on which queries can be executed.

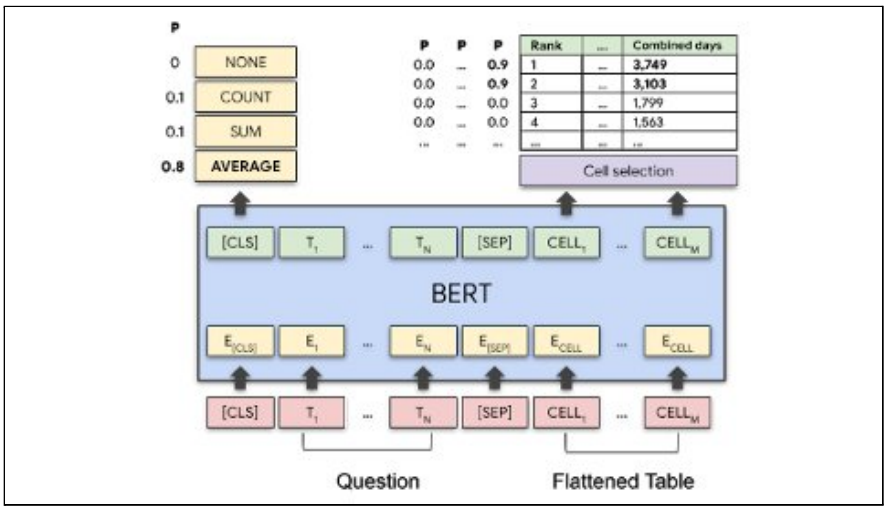


Figure 6.3 A bert based model - Tapas to perform table question answering

3. Result Estimation Module

The output of the image module would be a list of (element_type, confidence score, bounding box vector). The length of the list indicates the number of elements in the input plot. Now that bounding box annotations are obtained, next the points of references to the graph elements is passed along with the image to the OCR stage, in which the localized text content is extricated, after which we have utility modules to map the complete image data into its tabular equivalent and generate the csv, now things become simpler because the plot data is no more an image but it is a semi-structured table on which table question answering can be performed, just as in case of the wikitables questions dataset. The questions that are handled in our work are related to data-retrieval following a simple select, project or select cum project vice-versa, statistical mean, min - max, boolean truths, range based queries, simple summations, summations across an interval from the graph data, trend wise queries, differences with few constraints and restrictions laid on them.

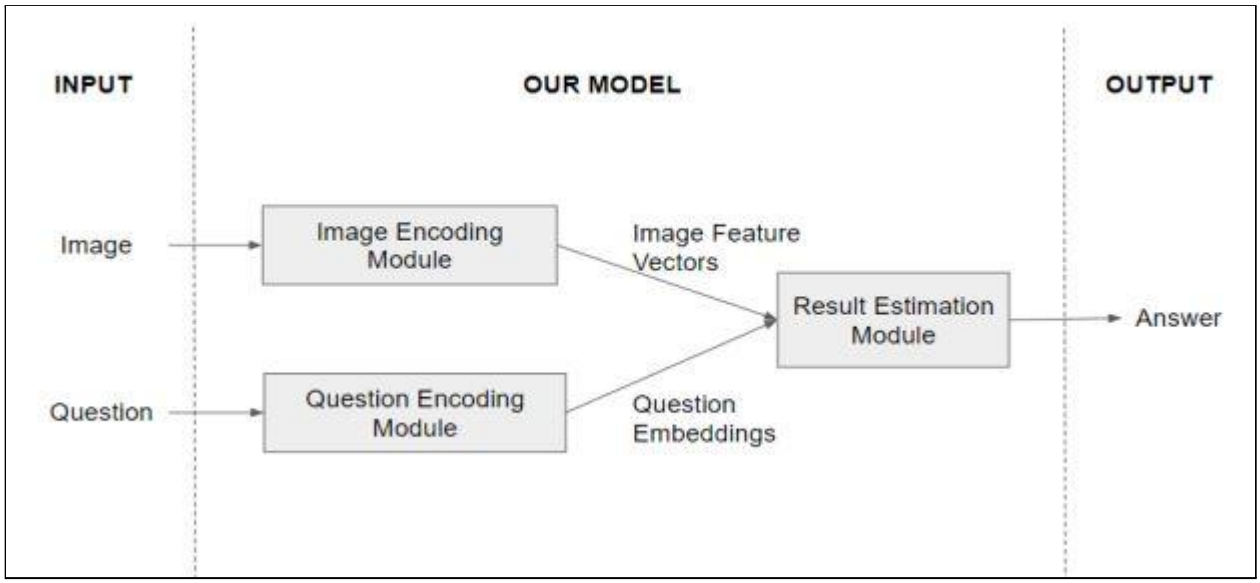


Figure 6.4

6.2 Low Level Design:

The Section deals with the lowest level dissection of the high level design developed in Phase 1. Here we delve into the modular and unit components that constitute the development of this tool.

6.2.1 Overview

The below diagram Summarizes the high level design that we intended to do.

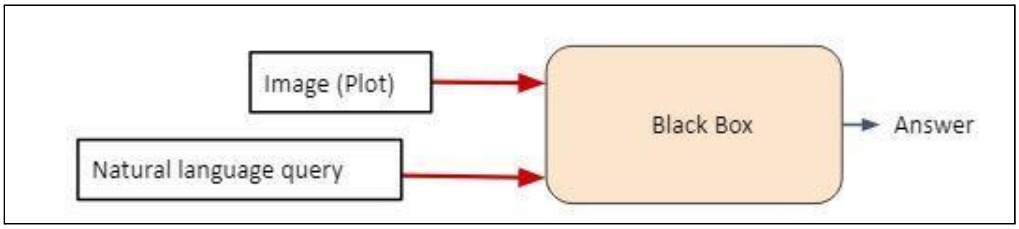


Fig 6.5 : Proposed High level View of The VQA system

In the figure we have two inputs being fed into the Black Box which are an input image [the graphical plot in scope] and a Natural language query related to the graphical image. whereas in the lower level design the black box is further expanded and cut open. The Black Box Constitutes the core of the visual question answering system.

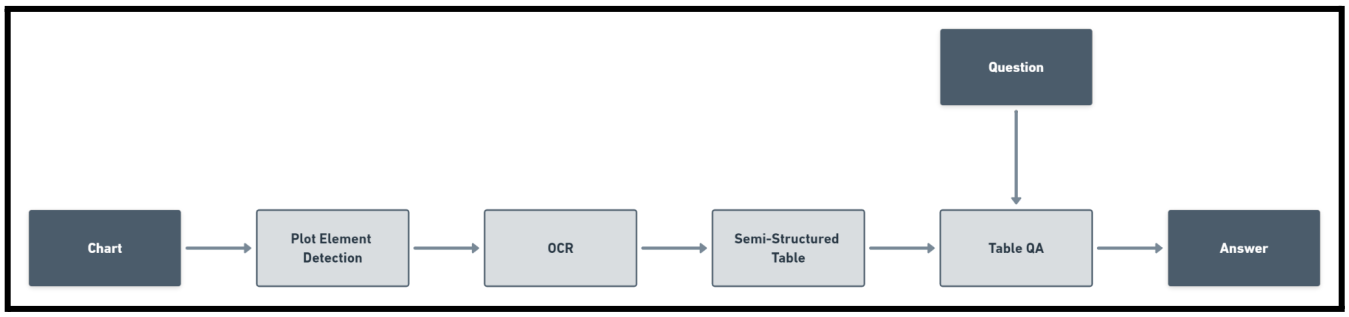


Fig 6.6 The cut open view of the black box and deep delve into the modules

There are 4 stages within the Black Box as shown in ***Fig 6.6***

- ➢ **The plot element detection stage**
- ➢ **The optical character recognition stage**
- ➢ **The Semi-structured table generation stage**
- ➢ **Table Question Answering Stage**

The inputs to black-box / VQA system

- A graphical plot [bar {vertical/horizontal/grouped}, dot, line]
- A question posed [in vocab / out of vocab] related to the plot image

The phase-wise dissection will be done in the following sections.

6.2.2 Purpose

The purpose of the low level design document is to provide a detailed description about the working of each and every unit and how they all work in union to achieve the desired result. It is used by designers, operation teams, implementers/dev and dev-test members. This will serve as a documentation for developing stubs/drivers.

Here we provide a description about the four stages within the Black Box and how they inter-communicate with each other and interface with each other. Moreover, this is the phase of a project in which the application logic is designed and ready to be implemented. The exit criteria / input criteria - data to every phase needs to be dissected and presented in detail.

6.2.3 Scope

The scope of this subchapter is to address the flow of information through the pipeline and stage-wise requirement which is needed to accomplish the goals of corresponding stages. This implementation of VQA on statistical plots takes into consideration **plots of type = {Dot, Line, Bar [Hbar, Vbar, Grouped]}** and **Questions = {Open-ended, In-vocabulary}.**

Overview of the tools that have significant contribution in every stage is provided. Alongside constraints, dependencies and assumptions existing between the modules/stages is also discussed. An overview regarding the novel practices and new ideas infused within the system is also discussed along with examples and variants of those.

6.2.4 Design Constraints, Assumptions, and Dependencies

The Environment, hardware software dependencies needed to run this pipeline

The training environment specification is as follows

- **Platform** : Google Colab Pro (Cloud)
- **RAM** : 26GB
- **Disk** : 110GB (Cloud)
- **GPU** : 16GB , Tesla - T4
- **Training Data Size** : 6.x GB
- **Test Data Size** : 1.5xGB

Assumptions of the model / VQA system

- ➢ The VQA is limited to only certain class of graphs and questions
- ➢ The input image to the model should be one among **{Dot, Line, Bar [Hbar, Vbar, Grouped]}**
- ➢ The questions posed should be of type **{Open-ended, In-vocabulary}.**

- The type of questions can be based on arithmetic mean, median, difference, sum, data retrieval, comparison or boolean.

Structural questions are not within the scope of this implementation.

Dependencies between the stages and the Input/Exit data+Criteria

Stage	Input Criteria	Exit Criteria	Output
Plot element detection	Input Plot Image belonging to certain class of graphs	Bounding box annotation around the plot elements.	A text tabular formatted equivalent of a json file , holding the coordinates of the plot elements[topleft_x , topleft_y,bottom_x , bottom_y] and the confidence that the element belongs to a class
OCR	text tabular file from previous stage + Image	use OCR module like tesseract to read the character within the bounding coordinates in the image	extracted texts from the bounding box specific region according to the category of the plot element
Semi-structured table generation	textual data extracted from OCR	format the data into a semi-structured table on which queries can be executed	A CSV file which is a tabular format of the graph input image

Table Question Answering	The CSV file/tabular format of the graph + Question	classify the queries into boolean vs data-retrieval Execute the queries on the table and accumulate answer	The final answer to the input question

Table 6.1

We can clearly see the **linear dependency that exists across the modules**; failure of any one of the intermediate modules will tend to have a cascading effect on the follow up stages.

There also exist few dependencies on the modules/off-the-shelf components that are being made use of in every stage.

Dependencies on the Modules/libraries and packages used

- Detectron-2 [5]
- Pytorch - 1.9.0
- Tesseract [6] , conda environments
- cv2, TAPAS [7], TABFACT & SEMPRE
- All other utility modules like OS, JSON, NUMPY, CSV, Scipy etc.

Constraints regarding the questions posed

Data retrieval, sum, average, columnar max- min questions handling capability was already built into TAPAS whereas we have added custom methods/operators handling methods to accept a variety of questions based on range, quartile, difference etc. These are based on break up over a particular keyword.

6.2.5 Module Wise Design Description

A) Plot Element Detection Stage

- Input : Image{graphical plot}
- Output : formatted text file derived from JSON

- The plot elements of an input image are extracted by training an object detection model over a large collection of samples {images + annotations}.
- **Detectron-2** is faster, flexible and vast in terms of configuration, models and implementation due to its API availability when compared with its parent, hence we use it as the object detection and bounding box generation tool.
- Few samples are shown below

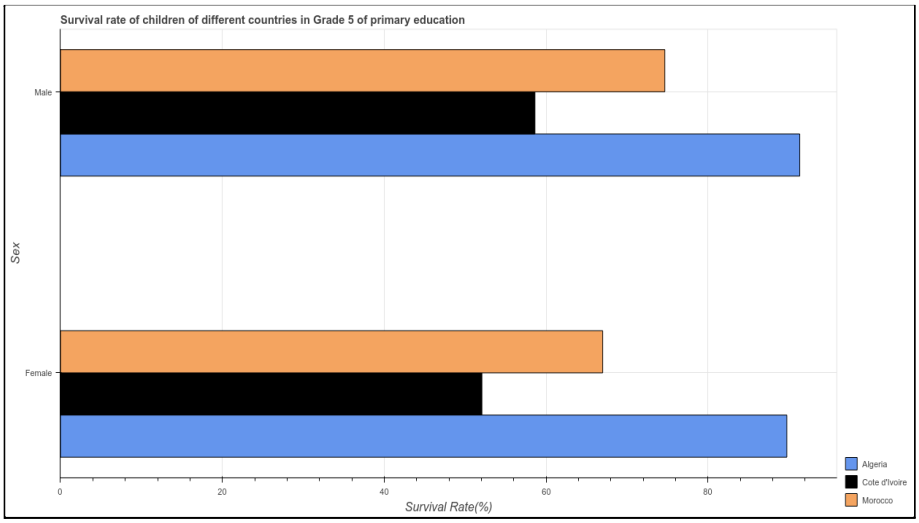


Figure 6.7 A horizontally grouped bar graph {Test-input}

The image along with its annotation is passed onto a model trained by us, we now have the weights of the model saved for further testing and inference, as the image passes through the designed object detection model, we expect bounding boxes to be drawn around every potential plot element which was learnt by the model. The output of a model will be a json file which maps all the detected objects (plot elements) to the **class** to which it belongs, with an appropriate **confidence value** and its **bounding box tensor**.

```
{'instances': Instances(num_instances=22, image_height=650, image_width=1245, fields=[pred_boxes: Boxes(tensor([[  27.2519,  459.9764,   63.6584,  474.0742],
        [  40.1922,  102.9239,   64.2441,  117.1245],
        [1173.8287,  619.1683, 1215.5663,  639.1750],
        [  74.6989,  109.7839,  726.3925,  163.6040],
        [ 735.3030,  612.0875,  747.5630,  625.9379],
        [  79.2105,  521.1633, 1080.1464,  574.4030],
        [  75.5007,  467.3218,  653.8602,  521.5388],
        [  79.1654,   56.3122,  898.4966,  110.0806],
        [  76.8082,  414.3173,  815.8287,  467.9611],
        [1174.1018,  596.1894, 1235.4202,  616.2552],
        [1173.9548,  572.9149, 1208.1724,  592.9194],
        [1148.6736,  572.9440, 1168.8115,  593.0646],
        [1148.6539,  619.1230, 1168.7286,  638.9912],
        [  76.3105,  164.4282, 1101.3408,  217.3091],
        [ 512.8422,  611.9535,  524.9588,  625.9399],
        [ 290.6320,  612.0994,  303.0508,  626.0181],
        [ 546.7985,  629.4926,  666.7787,  647.7834],
        [  72.1504,  612.0005,   78.2134,  625.9789],
        [ 956.0478,  612.1361,  968.4012,  626.0216],
        [1149.1019,  595.9470, 1169.0248,  616.4178],
        [  76.6021,    9.0043,  769.3833,   27.0149],
        [   6.1733,  301.8087,   24.0409,  330.2988]], device='cuda:0')), scores: tensor([1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 0.9999,
        0.9999, 0.9999, 0.9999, 0.9999, 0.9999, 0.9999, 0.9999, 0.9999, 0.9999,
        0.9999, 0.9999, 0.9997, 0.9996], device='cuda:0'), pred_classes: tensor([9, 9, 2, 0, 7, 0, 0, 0, 0, 2, 2, 4, 4, 0, 7, 7, 6, 7, 7, 4, 5, 8],
       device='cuda:0'))])}
```


Figure 6.7 The json output produced by detectron tester , which maps object elements to its class , with a certain confidence and the bounding box tensor

Given the classes to which the plot elements belong span from [0-9], we need a mapping between the class numbers and the plot elements, which is provided in Figure ***6.8*** below.

```
mapping = {  "bar": 0,
             "dot_line": 1,
             "legend_label": 2,
             "line": 3,
             "preview": 4,
             "title": 5,
             "xlabel": 6,
             "xticklabel": 7,
             "ylabel": 9,
             "yticklabel":9
          }
```

Figure 6.8

Now that we have the mapping and the unstructured json data, it is better to have them both combined and produce a readable and easily understandable textual format that can be used for further processing in the pipeline.

```
class confidence       top_x            top_y             bottom_x          bottom_y
bar 0.9999309778213501 76.31050872802734 164.42819213867188 1101.3407821655273 217.30908203125
bar 0.9999496936798096 76.8082275390625 414.31732177734375 815.82861328125 467.9610595703125
bar 0.9999642372131348 74.69889831542969 109.78387451171875 726.3924407958984 163.60403442382812
bar 0.9999502897262573 79.16533660888672 56.312198638916016 898.4965744018555 110.08055877685547
bar 0.9999561309814453 79.21051025390625 521.163330078125 1080.1463623046875 574.4030151367188
bar 0.9999512434005737 75.50065612792969 467.32183837890625 653.8601531982422 521.538818359375
legend_label 0.9999395608901978 1173.954833984375 572.9148559570312 1208.17236328125 592.91943359375
legend_label 0.9999446868896484 1174.101806640625 596.1893920898438 1235.420166015625 616.2551879882812
legend_label 0.9999701976776123 1173.8287353515625 619.1683349609375 1215.5662841796875 639.175048828125
preview 0.9999374151229858 1148.673583984375 572.9439697265625 1168.8115234375 593.0646362304688
preview 0.999933123588562 1148.6539306640625 619.1229858398438 1168.7286376953125 638.9912109375
preview 0.9998660087585449 1149.1019287109375 595.9469604492188 1169.0247802734375 616.4177856445312
title 0.9996670484542847 72.77204246520996 8.644117126464844 775.538344116211 28.365630197525025
xlabel 0.9998942613601685 519.4585968017578 629.4925537109375 672.112916015625 647.783447265625
xticklabel 0.9998867511749268 72.15039825439453 612.00048828125 78.21342468261719 625.9788818359375
xticklabel 0.9999247789382935 512.8422241210938 611.9534912109375 524.9588012695312 625.9398803710938
xticklabel 0.9999061822891235 290.6319580078125 612.0994262695312 303.05084228515625 626.0181274414062
xticklabel 0.9999575614929199 735.3030395507812 612.0874633789062 747.5630493164062 625.9378662109375
xticklabel 0.9998866319656372 956.0477905273438 612.1361083984375 968.4012451171875 626.0216064453125
ylabel 0.9995629191398621 5.8646334409713745 289.736396484375 24.233238487243653 346.813737487793
yticklabel 0.9999923706054688 40.192203521728516 102.92390441894531 64.24410247802734 117.1244888305664
yticklabel 0.9999945163726807 27.25185203552246 459.97637939453125 63.65839195251465 474.07421875
```

Figure 6.9 : properly formatted text file which has the bounding box tensors of all the plot elements that were detected in the model

As seen above, all the bounding boxes corresponding to the plot elements have been extracted according to their classes. We chose **Faster-RCNN** as our object detection model which is a **descendant from the R-CNNs series**. The heuristic behind doing so is due to the presence of the RPN (Regional proposal network is known for locating feature targets accurately) and the inference time.

We also need to decide on the **feature extractor model** for serving as the **backbone / feature-extractor** for our faster-RCNN Setup. We Chose **Resnet-101** to be our feature extractor due to **skip connections** and **residual blocks** which can reduce the problem of vanishing gradients in a very deep network like Resnet-101.

B) OCR detection Stage

- The textual format of the json file as shown in ***Figure 6.9*** is passed into this stage.
- The images directory is made accessible to this module.
- With the help of bounding box coordinates extracted, we can accurately and locally capture the text information within the bounding boxes rather than passing an entire image into the OCR module.
- The captured text is then read using OCR and classified into its category.

C) Semi Structured table generation

This phase is the crucial phase of converting the graphical data to its tabular format based on the OCR readings done in accordance with classes. We'll walk through an example to explain the conversion of an image (plot) into a tabular data format (csv). An input graph goes through several transformations in the pipeline but not necessarily in the image format; right after the OCR stage, the input image is no longer required, the image is discarded and its tabular csv format comes into play for the final table QA stage.

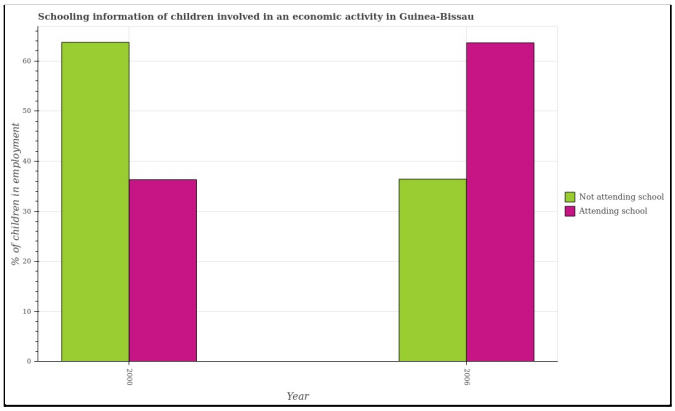


Figure 6.10 : A Simple Bar Graph

The simple bar graph goes through the plot element detection stage, and a text file emerges as the outcome as discussed in the previous stage. This text file will then be passed onto the OCR stage along

with the image to extract the textual content within the boxed coordinates. Following which the information is structured in the semi structured table generation phase.

```
Category   Score             left_x            top_y             right_x           bottom_y
yticklabel 0.9999904632568359 27.946645736694336 246.9253387451172 41.919342041015625 260.8976135253906
yticklabel 0.9999895095825195 27.873197555541992 328.9063720703125 41.86872100830078 343.05303955078125
bar 0.9999873638153076 640.291259765625 284.0574645996094 749.989990234375 580.9351806640625
bar 0.9999823570251465 201.92819213867188 281.8130798339844 312.2918701171875 578.4205322265625
legend_label 0.9999779462814331 932.5986328125 303.1197509765625 1071.5126953125 322.6226806640625
yticklabel 0.9999773502349854 27.868160247802734 409.8987121582031 41.989967346191406 423.9773254394531
bar 0.9999696016311646 750.8607177734375 62.46775817871094 861.982421875 581.0493621826172
xticklabel 0.9999669790267944 194.113037109375 590.9034423828125 208.31503295898438 618.8867797851562
yticklabel 0.9999643564224243 34.982704162597656 572.9815063476562 41.99324035644531 587.0731201171875
bar 0.9999592304229736 91.97103881835938 59.920265197753906 202.5250701904297 581.5347671508789
preview 0.9999498128890991 908.1177368164062 303.0318908691406 928.0110473632812 322.7835998535156
yticklabel 0.9999486207962036 28.012062072753906 164.76251220703125 42.00925827026367 178.90281677246094
yticklabel 0.9999397993087769 28.047203063964844 492.2265625 42.02458572387695 505.870849609375
xlabel 0.9999388456344604 457.864501953125 627.0880737304688 493.36260986328125 646.0440673828125
xticklabel 0.999916672706604 743.106201171875 590.95849609375 756.9241943359375 619.1182861328125
legend_label 0.9998894929885864 932.1422119140625 325.8930969238281 1043.439697265625 345.8262023925781
preview 0.9998648166656494 908.1642456054688 326.23406982421875 928.19921875 346.0467834472656
yticklabel 0.9998058676719666 28.096742630004883 84.25582885742188 42.00232696533203 97.83050537109375
ylabel 0.9998049139976501 6.735439777374268 190.0906524658203 26.02379274368286 425.5273742675781
title 0.9994369149208069 49.137367248535156 9.029414176940918 796.2882461547852 27.020319938659668
```

Figure 6.11 : Textual output of input graph

	Year	Attending school	Not attending school
0	2000	37.06615560158244	64.78018619662012
1	2006	64.48456969920525	37.0474010349917

Figure 6.12 : Tabular CSV format of the input graph

Now this csv file would be further used in the table question answering phase.

D) Table Question Answering Stage

- We have made use of **Google's TAPAS** to answer questions from tabular data
- Tapas selects a subset of table cells and applies aggregation/retrieval operations on top of them
- It extends BERT architecture with additional embeddings that capture tabular structure, and with two classification layers for selecting cells and predicting a corresponding aggregation operator.
- We have added our custom operations and methods to suit the desired output.
- It is trained on Wikipedia Tables and provides a pre-trained model for end tasks.

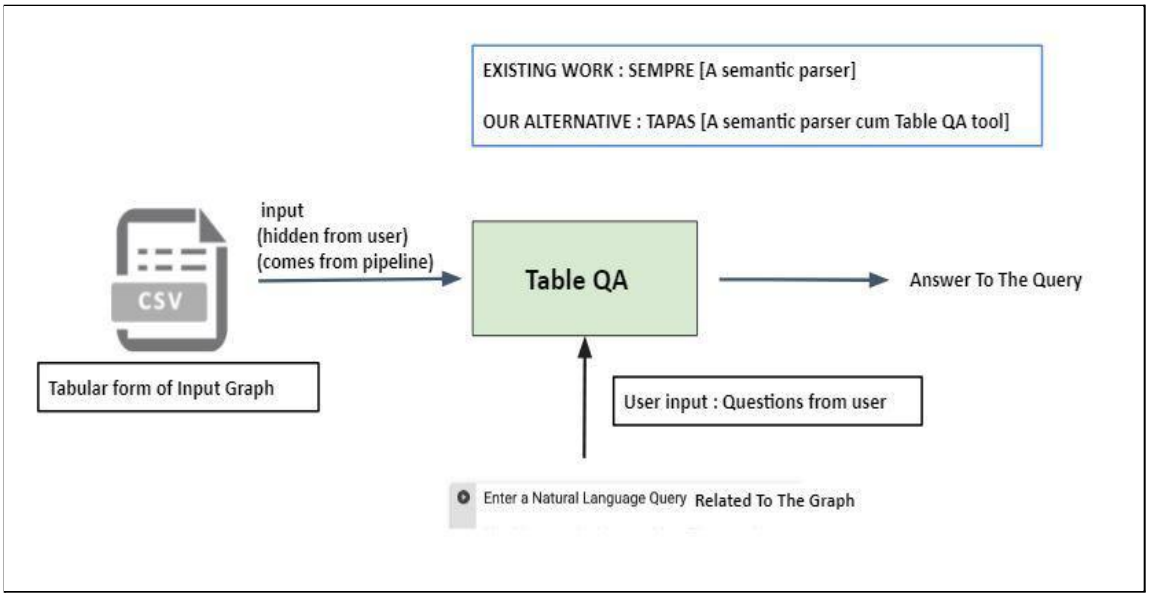


Figure 6.13 : The Table Question Answering Stage

CHAPTER 7

Proposed Methodology

The following sections comprise the detailed idea about the working of the pipeline designed for this work. It consists of the novelty induced by us in this work, the inspiration for this work and the key aspects of the model.

Stage 1 : The Plot Elements Detection

- From the previous sections it is certain that the primary step is to detect the possible plot elements, every single element needs to be captured, hence we adopt an object detection tool called the detectron, we choose the successor of the detectron, which is detectron-2
- Detectron-2 has a vast collection of models in its model zoo which provides us a flexible option to choose the baseline and backbone networks at ease by keeping the speed/accuracy trade-offs in consideration.
- The input dataset is a collection of training images and their corresponding annotation file; they reside separately. They must be correlated together before they are ready for training. This is easily achieved by detectron-2's dataset catalog and metadata set catalog.
- The choice of the model has been discussed in the previous system design section. The output of this stage is a json file which is converted into a structured text file.

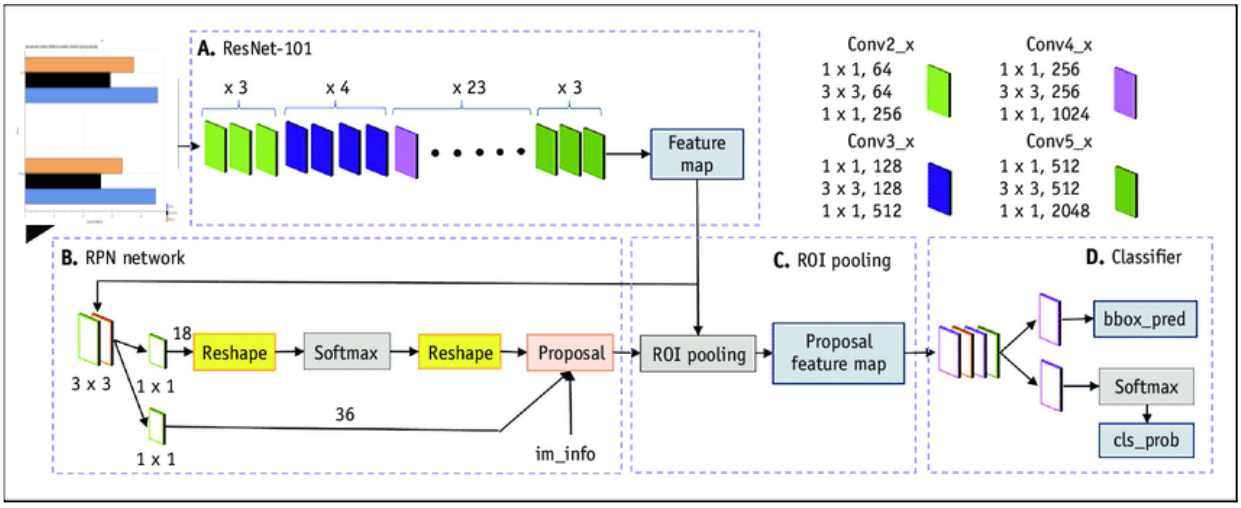


Figure 7.1 The illustration of an image being fed into a faster-RCNN with Resnet 101 backbone

Given the classes to which the plot elements belong span from [0-9], we need a mapping between the class numbers and the plot elements, which is provided in ***Figure 6.8*** **.**

Stage 2 : The Optical Character Reader/Recognition Stage

- There are a total of 10 different plot elements that can be found in any statistical plot. These can be grouped into two categories: Textual Elements and Visual Elements
- Textual elements correspond to the title of the plot, y-axis label, x-axis label, x-tick, y-tick values and the labels corresponding to the legend
- To read the textual and numeric data off the textual components, we make use of the formatted textual output from the previous stage, and an Optical Character Recognition module
- The OCR module used is pyocr which is a wrapper for the Tesseract OCR engine
- Detected textual elements are cropped to the bounding box size (which is obtained from the previous stage), then converted to gray-scale and passed onto the pyocr module.
- Thus, the output of this stage is textual data corresponding to the detected textual elements.

Stage 3 : Semi Structured table generation

- The output of this stage is a semi-structured table that encapsulates all of the data in the statistical plot.
- For the textual elements, we have already obtained textual data. This stage is responsible for mapping legend values to the legend color, x-ticks to the x-axis label and the y-ticks to the y-axis label. This is done by associating the legend / x-tick / y-tick value bounding box to the closest legend color / x-axis / y-axis boundary respectively.
- For the visual elements, each element is associated with an axis, and a corresponding legend. The color of the visual element is matched with the legend colors, and the legend of the closest match is associated with the element. To find the value associated with the bar, the information of height is taken from the bounding box representation, and the closest y-tick is mapped.
- Doing this for all visual elements will fill the table.

Stage 4: Table Question Answering Stage

- Given a semi-structured table and a relevant natural language question as input, this stage is responsible for producing an answer to the question from the table as an output.
- The questions can be classified into two types. The first type corresponds to open-ended questions that have an unrestricted answer domain. The second type corresponds to questions that require a Yes/No (binary) answer.
- To handle open-ended questions, we have made use of the existing TaPas (Table Parsing) model. This model is based on the BERT's encoder with certain modifications. Positional embeddings are used to encode tabular data, and two additional classification layers are introduced to select cells of the table and the aggregation operation to be performed.
- Our work makes use of a pre-trained TaPas model that has been trained on the WikiTables Questions dataset with intermediate pre-training. This model can handle 3 types of aggregation operations - SUM, COUNT, AVERAGE. To add to the capabilities of this model, we have added other operations such as RATIO, DIFFERENCE, MEDIAN, TREND, RANGE and QUARTILES.
- To handle questions that require a Yes/No answer, we have used a TaPas model trained on the TabFact dataset. This is a dataset used for table entailment and fact verification. We have extended its capabilities by adding other operations like in the earlier mentioned model.
- An important aspect here is that given an input question we would need to know what type of question it is (whether it is an open-domain question or a yes/no question). For this, we have implemented a binary question classifier.

Binary Classifier

- There are two categories of questions addressed: Open-ended and Yes/No. Each of these is an independent model and hence to integrate into a single pipeline, a binary classifier is used.
- Binary classifier model classifies the given input question into Yes/No class (class 0) or Open-ended class (class 1)
- A dataset is prepared for this purpose from the PlotQA data. The model is trained on questions of all types of plots (i,e, vbar_categorical, hbar_categorical, dot_line, line) and the answers which are converted into the categories of 0 or 1.

- The Deep Learning model is trained to classify the input questions into correct classes. The following are the model specifications:
  - Embedding Type: GloVe
  - Max Pooling Layer: 1
  - Dense Layer: 10 nodes and ReLu activation
  - Dense Layer: 1 node and Sigmoid activation
  - Optimizer: Adam
  - Loss Function: Binary Cross-Entropy
  - Metric: Accuracy

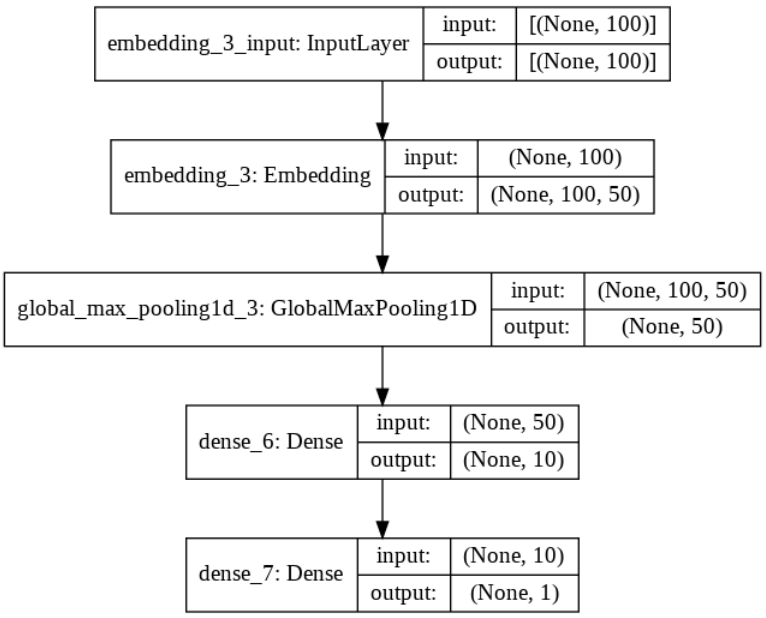


Figure 7.2

- The model was trained for 10 Epochs in a batch size of 10. Accuracy of 1.0 was obtained.

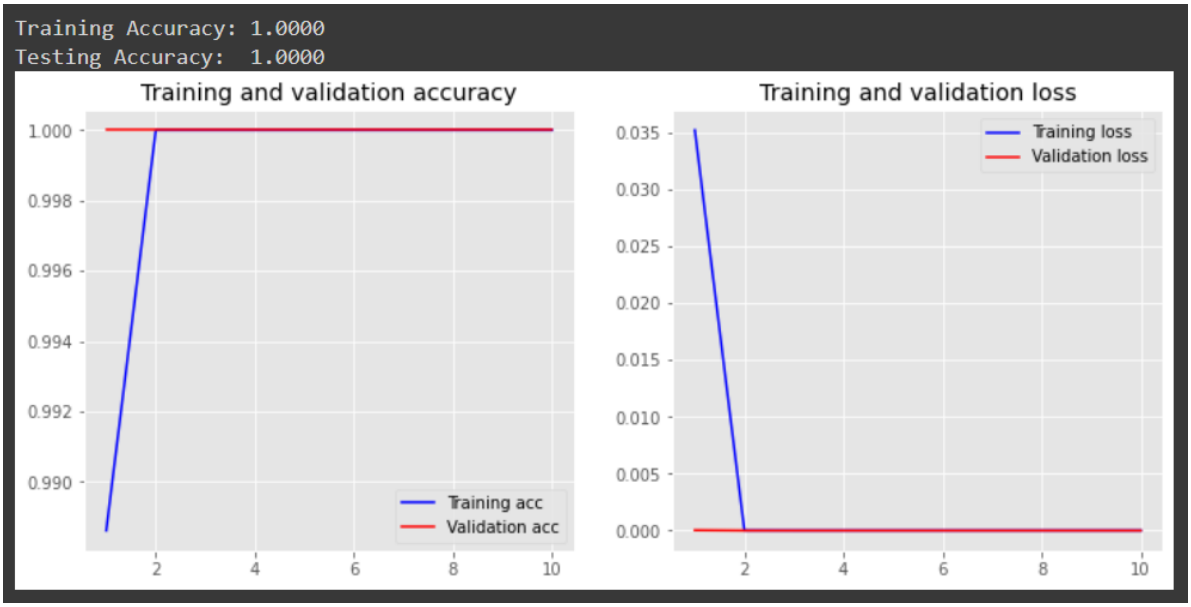


Figure 7.3

- The trained model was saved and loaded to classify test images. If the model outputs class 0, the question is passed to TABFACT model and if the model outputs class 1, the question is passed to TAPAS model.

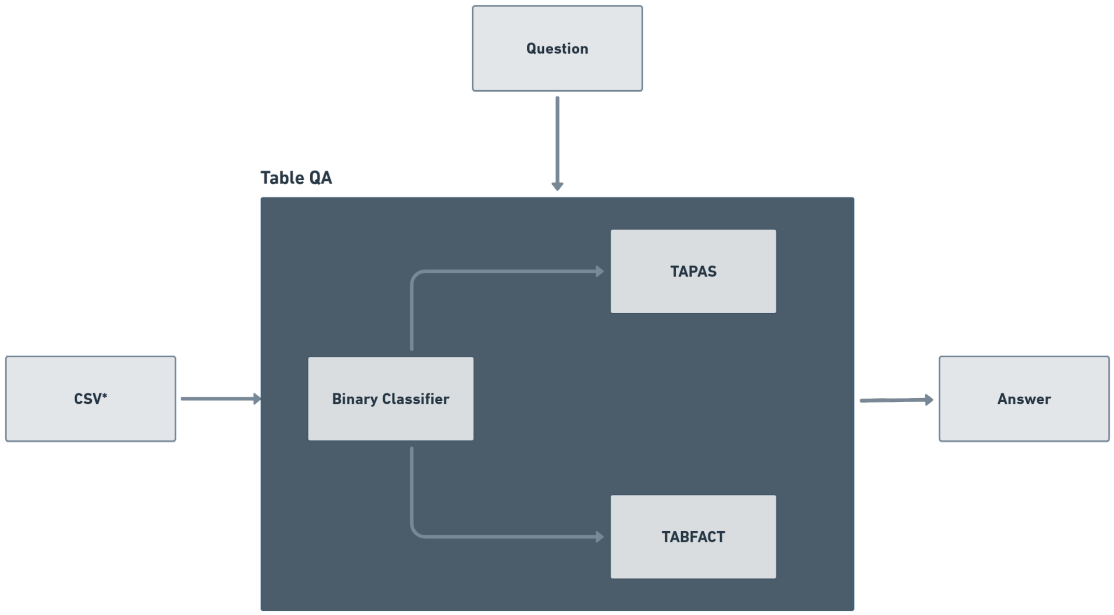


Figure 7.4 Table Question Answering Model

CHAPTER 8

Implementation and Pseudocode

This section details the implementation details and a pseudocode for each of the modules in our pipeline.

Stage 1 : Plot Elements Detection

To detect the elements in the plot and map it to its bounding box representation, we have made use of Detectron 2 - an object detection tool with a bounding box generator.

The training data includes the training images and the annotations from the PlotQA dataset. We correlate the image with its corresponding annotation and pass it to the Detectron trainer. We chose the object detection model to be the Faster-rcnn and Resnet_101 as the backbone network. The model was trained for 2 lakh iterations on a learning rate of 0.0004, with a decay of 0.1.

On all of the test images, the model was tested to generate the bbox representations as a JSON file, which was then formatted to a text file.

```
#TRAINING

Load (train_images , Annotation_file)
model = Trainer()
dataset : Split(train_images , Annotation_files , Splits = 3)
model.data : Correlate_dataset_with_annotations(dataset)
sample(data) #check If Image And Annotation Are Corresponding
choose Object_detection_model And Backbone_network
model.object_detection_model : Faster-rcnn
model.backbone_model : Resnet_101
iters : 21 ; Lr : 0.0025 ; Gamma : 0.1
model.train(data , Iters , Lr , Gamma Object_detection_model , Backbone)

#TESTING

Load(test_images)
model.test_data :Register_for_testing(test_images)
json_result : model.test(generate_json_result = True)
txt_format : convert_json_o_text(json_result)

```

Figure 8.1 Pseudocode of the Plot Elements Detection Stage (Detectron - 2)

```
from detectron2.data.datasets import register_coco_instances
#split-1
register_coco_instances("Train_A", {}, "/root/Work/Data/PlotQA/annotations/train_50k_annotations.json", "/root/Work/Data/TRAIN/png")
#implies that images from the specified path and the annotation from the specified path must be correlated and superimposed before training

#split-2
register_coco_instances("Train_B", {}, "/root/Work/Data/PlotQA/annotations/train_50k_1l_annotations.json", "/root/Work/Data/TRAIN/png")

#split-3
register_coco_instances("Train_C", {}, "/root/Work/Data/PlotQA/annotations/train_1l_end_annotations.json", "/root/Work/Data/TRAIN/png")
```

Figure 8.2 Registering The Data Splits For training (Image+Annotation{json})

```
config.merge_from_file(model_zoo.get_config_file("COCO-Detection/faster_rcnn_R_101_FPN_3x.yaml"))

config.DATASETS.TRAIN = ('Train_A' ,'Train_B' , 'Train_C')

config.DATASETS.TEST = ()

config.NUM_GPUS = 1

config.DATALOADER.NUM_WORKERS = 4

config.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-InstanceSegmentation/mask_rcnn_R_101_FPN_3x.yaml")   # Let training initialize from model zoo

config.SOLVER.BASE_LR = 0.0004                                          #kept minimal , so that global optimum can be hit during gradient descent

config.SOLVER.MAX_ITER = 200000                                          #Must Change for future training , similar to epoch

config.SOLVER.STEPS = (1100,40000,120000)                                          #Stages at which LR Reduction must occur

config.SOLVER.GAMMA = 0.1                                               #Reduction factor for LR

config.SOLVER.IMS_PER_BATCH = 1                                         #Images seen by GPU per sec

config.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 512                       #Batch-size

config.MODEL.ROI_HEADS.NUM_CLASSES = 11                                 #10 instances + 1 background

config.OUTPUT_DIR = './Output_2L_Cont'

#config.TEST.EVAL_PERIOD = 30000

config.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.7
```

Figure 8.3 The training yaml configuration file customized as per our LLD

```
from detectron2.engine import DefaultTrainer

trainer = DefaultTrainer(config)
trainer.resume_or_load(resume=False)
trainer.train()
```

Figure 8.4 The training Stage

Stage 2 : The Optical Character Reader/Recognition Stage

We used the pyocr tool (a wrapper for the Tesseract Engine) for optical character recognition. This was used for all the textual elements detected in the previous stage.

```
22  ------------------------------------------------------------------------------
23
24
25  ocr : OCR()
26  ocr.load_utilities()
27  ocr_extractions : ocr.load(txt_format  , input_image)
28
29
30  ------------------------------------------------------------------------------
```

Figure 8.5 Pseudocode of the OCR Stage

Stage 3 : Semi Structured table generation

The output of this stage is a semi-structured table in the form of a CSV. The textual and numeric information extracted from the ocr is populated into a table. The values (height information in the case of a bar plot, x and y coordinates in case of a line and dot-line plot) are filled using the logic mentioned in the proposed methodology section.

```
CSV_maker : CSV()
CSV_maker.ocr_readings : ocr_extractions()
Tabular_format : CSV_maker.generate_table_csv()
```

Figure 8.6 Pseudocode of Semi-structured Table Generation

Stage 4: Table Question Answering Stage

We have made use of the existing TaPas (Table Parsing) module for the task of question answering from tables. The csv obtained from the previous stage is pre-processed. This is the stage where the

questions entered by the user are processed. As mentioned earlier, the questions are classified into either an open-ended question or a question that requires a Yes/No answer. If the question is an open-ended question, it is passed onto the TaPas model pre-trained on the WTQ dataset, otherwise it is passed on the TaPas model trained on the TabFact table entailment dataset. The output here is an answer to the question.

```
table_qa : TAPAS()
csv_table : preprocess(Tabular_format)
table_qa.table : csv_table
table_qa.queries : List(Read_from_users())
table_qa.queries.classify()
table_qa.answers()
```

Figure 8.7 Pseudocode of Table Question Answering

AN END-TO-END EXAMPLE

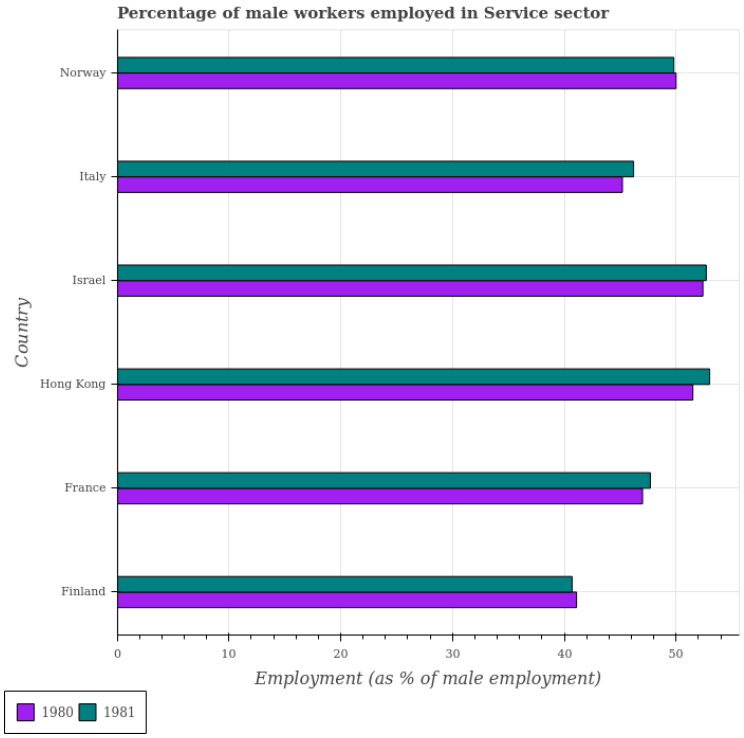


Figure 8.8 Input Image

Input : An image and a question

Output : An answer to the question

Given the input image as shown in ***Figure 8.8***, we demonstrate each of the stages below.

Stage 1 : The Plot Elements Detection

The output of this stage is a formatted text file that contains the bounding box representations of each of the elements of the plot represented in ***Figure 8.9***. This statistical plot contains a total of 31 plot elements. Therefore, the output shows a txt file with 31 lines, each corresponding to a plot element.

5574.txt

```
1 yticklabel 0.9999946355819702 70.65868377685547 166.0301055908203 95.90317535400039 180.24429321289062
2 yticklabel 0.9999940395355225 56.70259094238281 460.2558288574219 96.12216186523438 473.992431640625
3 yticklabel 0.9999926090240479 52.40139389038086 68.11477661132812 96.11300659179688 81.90919494628906
4 yticklabel 0.9999892711639404 64.40562438964844 264.47857666015625 96.38269805908203 278.0016174316406
5 bar 0.9999853372573853 107.04766845703125 565.3331298828125 539.787109375 580.1272583007812
6 bar 0.999968409538269 106.83328247070312 74.97222900390625 635.513916015625 89.83660125732422
7 preview 0.9999650716781616 10.059914588928223 668.806884765625 29.93454360961914 688.8609619140625
8 bar 0.9999635219573975 108.80455017089844 354.7571105957031 669.4835205078125 369.15228271484375
9 yticklabel 0.9999619722366333 53.4467887878418 558.0203857421875 96.28006744384766 572.0387573242188
10 bar 0.9999618530273438 105.96603393554688 369.0704650878906 651.154052734375 384.01531982421875
11 preview 0.9999606609344482 69.05435180664062 668.9619750976562 88.89054870605469 689.03515625
12 bar 0.9999605417251587 106.07267761230469 270.9334411621094 661.7080688476562 285.9010314941406
13 bar 0.9999490976333618 107.0818099975586 467.1742248535156 604.5418701171875 482.14801025390625
14 bar 0.9999490976333618 107.93446350097656 256.6279296875 669.5387573242188 271.04522705078125
15 bar 0.9999490976333618 107.84508514404297 172.93421936035156 584.4448852539062 187.95010375976562
16 xticklabel 0.9999262094497681 523.00341796875 617.0694580078125 536.9373168945312 631.1053466796875
17 bar 0.9999213218688965 107.91783905029297 550.761962890625 536.0125732421875 565.3912963867188
18 legend_label 0.9999172687530518 34.85472106933594 668.88671875 65.8724365234375 689.089599609375
19 legend_label 0.9999128580093384 93.74087524414062 669.108642578125 124.3467025756836 689.2234497070312
20 xticklabel 0.9999068975448608 104.04006958007812 616.9739990234375 110.94221496582031 630.9584350585938
21 xticklabel 0.9999068975448608 204.91419982910156 617.0042114257812 218.94931030273438 631.0257568359375
22 xticklabel 0.9998940229415894 417.209716796875 617.04736328125 430.898681640625 631.0352172851562
23 xticklabel 0.9998908042907715 310.7530517578125 617.034423828125 325.072265625 630.994384765625
24 bar 0.9998842477798462 109.16693115234375 452.7776794433594 612.639892578125 467.078369140625
25 xticklabel 0.9998824596405029 627.995849609375 616.922119140625 642.1049194335938 630.9342041015625
26 bar 0.9998799562454224 107.85015869140625 60.325401306152344 634.2732543945312 75.0029296875
27 bar 0.999862790107727 109.2906494140625 158.4036865234375 593.994384765625 172.654296875
28 yticklabel 0.9998502731323242 33.24961853027344 361.7756042480469 96.37665557861328 376.0375061035156
29 ylabel 0.9997543692588806 7.122566223144531 286.8748779296875 26.106121063232422 352.16131591796875
30 xlabel 0.9997450709342957 238.1763916015625 637.1104736328125 565.7171020507812 656.1470336914062
31 title 0.9996367692947388 106.48019409179688 8.978804588317871 514.6640014648438 27.022794723510742
```

Figure 8.9 Output of Detectron - 2

Stage 2 & 3: The Optical Character Recognition and Semi Structured table generation stage

The bounding box representations are mapped onto the image to obtain textual and numeric information, which is then populated into a tabular format. ***Figure 8.10*** shows the CSV output obtained for the image in ***Figure 8.8***.

	Country	1980	1981
0	Finland	41.56915064054187	41.21788389031577
1	France	48.97010591234	48.40140328049439
2	Hong Kong	53.624025195560996	54.12660380966694
3	Israel	54.191569798758344	54.5127801772846
4	Italy	46.373520540205824	46.33205263996708
5	Norway	51.48881822230819	51.018932095367745

Figure 8.10 Output of Stage-3 (CSV)

Stage 4: Table Question Answering Stage

This stage makes use of the CSV as seen in ***Figure 8.10*** to answer the question posed by the user. The types of questions that can be addressed by our model are as follows.

1. **Count**

   Q: What is the total number of countries ?
   A: 6

2. **Sum**

   Q: What is the total number of male workers employed in 1980 ?
   A: 296.2171903097152

3. **Average**

   Q: What is the average number of male workers in the year 1980 ?

A: 49.36953171828586

Q: What is the average number of male workers in the year 1981 ?
A: 49.26827598218275

4. **Minimum**

Q: across all countries, what is the minimum percentage of male workers employed in the service sector in 1980 ?
A: 41.56915064054187

5. **Maximum**

Q: across all countries, what is the maximum percentage of male workers employed in the service sector in 1980 ?
A: 54.191569798758344

6. **Difference**

Q: What is the difference between the average number of male workers employed for the year 1980 and 1981 ?
A: DIFFERENCE = 0.10125573610311278

Q: What is the difference between the number of male workers employed for the country of France in 1980 and 1981 ?
A: DIFFERENCE = 0.5687026318456105

- Keyword = “difference between”
- The two entities must be separated by "and"

7. **Median**

Q: What is the median number of male workers employed in the year 1980 ?
A: MEDIAN = 50.22946206732409

- Keyword = “median“
- Column name for which the median has to be found

8. **Ratio**

Q: What is the ratio of male workers employed in 1981 to 1980 for the country hong kong ?
A: RATIO = 1.0093722657385955

- Keyword = “Ratio“
- QUANTITY_1 "to" QUANTITY_2

9. **Trend (Increasing or Decreasing)**

Q: What is the trend of male workers employed for the countries finland, france, hong kong in 1980 ?
A: TREND = INCREASING

Q: What is the trend of male workers employed for the countries israel, italy in 1980 ?
A: TREND = DECREASING

Q: What is the trend of male workers employed for the countries israel, italy, norway in 1980 ?
A: TREND = NONE

- Keyword = “trend“
- List of comma separated entries followed by "in" COL_NAME

10. **Selection operation on cell**

Q: What is the number of male workers employed for country france in the year 1981 ?
A: 48.40140328049439

11. **Select operation on cell after applying aggregation operation**

Q: Which country has the minimum number of male workers employed in the year 1981 ?
A: Finland

12. **Selection and Aggregation operation on subset of rows**

Q: What is the sum of male workers employed for the countries france, finland in the year 1980 ?
A: 90.53925655288188

Q: What is the average number of male workers employed for the countries france, finland in the year 1980?
A: 45.26962827644094

Q: What is the maximum number of male workers employed for the countries france, finland in the year 1980 ?
A: 48.97010591234

13. Project operation on column

Q: What are the names of all the countries ?
A: Finland, France, Hong Kong, Israel, Italy, Norway

Q: list out the countries
A: Finland, France, Hong Kong, Israel, Italy, Norway

14. Range

Q: What is the range of % of male employment for the year 1980 ?
A: RANGE = 12.622419158216474

15. Quartiles (Q1 and Q3)

Q: find the quartiles for the year 1980
A: FIRST QUARTILE (Q1) = 43.97133559037385
SECOND QUARTILE (Q2) = 50.22946206732409
THIRD QUARTILE (Q3) = 53.90779749715967

16. IQR

Q: find the interquartile range for the year 1980
A: INTER-QUARTILE RANGE = 9.936461906785823

17. Structural Query

Q: What is the title of the graph ?
A: TITLE OF THE GRAPH = Percentage of male workers employed in Service sector

Q: What is the label or title of the x-axis ?
A: X-LABEL = Employment (0S % of male employment)

Q: What is the label or title of the y-axis ?
A: Y-LABEL = Country

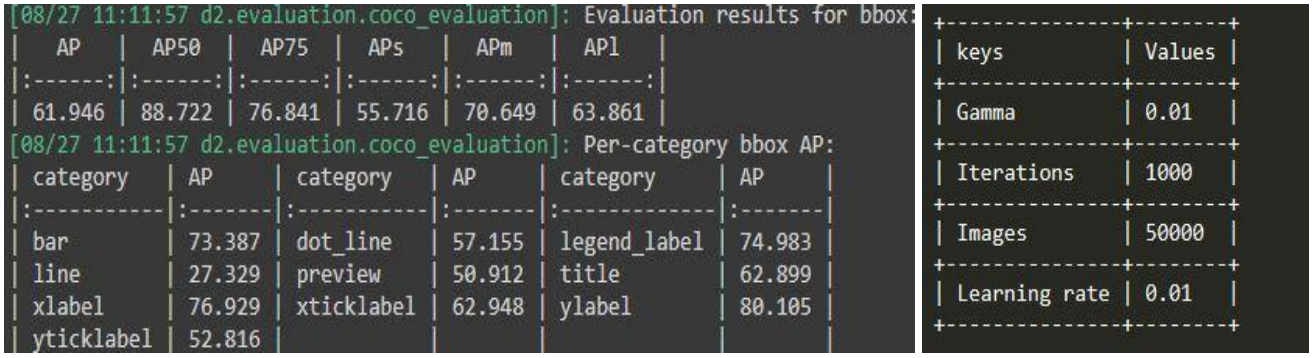
CHAPTER 9

RESULTS AND DISCUSSION

The steps mentioned in our methodology were followed and the results obtained by us are displayed in the following sections along with the heuristic behind how they have been obtained with supporting evidence.

9.1 Plot Element Detection Stage

The aim in this stage was to ensure that there was maximal overlap between the bounding boxes proposed by our object detection trained model with saved weight and the actual bounding box locations of the test image which are hidden from the model. AP aka Average precision is the score to look out for, lies between 0 - 100%, more the value of AP, higher the overlap and better the prediction made by the model. This again requires right selection of models, appropriate number of training iterations and Gamma factor. Below we display the results for various parameter settings and the incremental growth achieved by the model.



[08/27 11:11:57 d2.evaluation.coco_evaluation]: Evaluation results for bbox:

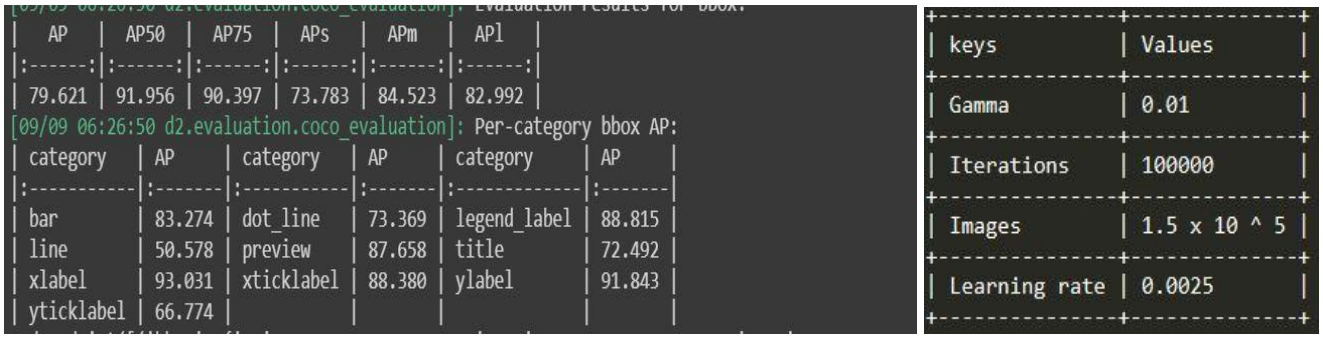
AP	AP50	AP75	APs	APm	APl
61.946	88.722	76.841	55.716	70.649	63.861

[08/27 11:11:57 d2.evaluation.coco_evaluation]: Per-category bbox AP:

category	AP	category	AP	category	AP
bar	73.387	dot_line	57.155	legend_label	74.983
line	27.329	preview	50.912	title	62.899
xlabel	76.929	xticklabel	62.948	ylabel	80.105
yticklabel	52.816				

keys	Values
Gamma	0.01
Iterations	1000
Images	50000
Learning rate	0.01

Figure 9.1 Test Accuracy After Trial - 1 With Parameter Setting on RHS



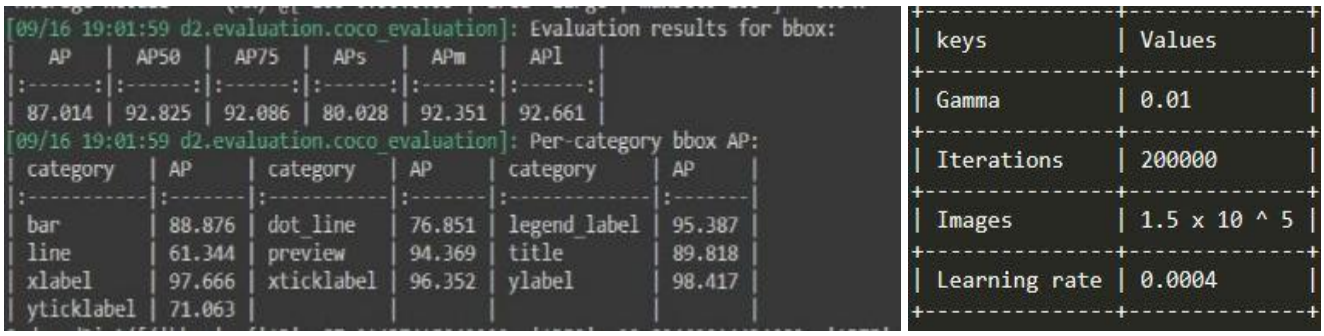
AP	AP50	AP75	APs	APm	APl
79.621	91.956	90.397	73.783	84.523	82.992

[09/09 06:26:50 d2.evaluation.coco_evaluation]: Per-category bbox AP:

category	AP	category	AP	category	AP
bar	83.274	dot_line	73.369	legend_label	88.815
line	50.578	preview	87.658	title	72.492
xlabel	93.031	xticklabel	88.380	ylabel	91.843
yticklabel	66.774				

keys	Values
Gamma	0.01
Iterations	100000
Images	1.5 x 10 ^ 5
Learning rate	0.0025

Figure 9.2 Test Accuracy After Trial - 2 With Parameter Setting on RHS



[09/16 19:01:59 d2.evaluation.coco_evaluation]: Evaluation results for bbox:

AP	AP50	AP75	APs	APm	APl
87.014	92.825	92.086	80.028	92.351	92.661

[09/16 19:01:59 d2.evaluation.coco_evaluation]: Per-category bbox AP:

category	AP	category	AP	category	AP
bar	88.876	dot_line	76.851	legend_label	95.387
line	61.344	preview	94.369	title	89.818
xlabel	97.666	xticklabel	96.352	ylabel	98.417
yticklabel	71.063				

keys	Values
Gamma	0.01
Iterations	200000
Images	1.5 x 10 ^ 5
Learning rate	0.0004

Figure 9.3 Test Accuracy After Trial - 1 With Parameter Setting on RHS

It can conclusively be seen that the AP increases linearly with the Iterations. Better training produces better results. The maximum number of iterations we could run was for 2 lakh iterations. Per category bbox AP value also tends to increase along with more training. This better AP helps to capture the textual and pictorial elements better in the further stages of the pipeline.

9.2 Table Question Answering Stage

The Table QA Stage outputs an answer for the question using the CSV generated from the semi-structured table generation stage. The accuracy of this stage depends on the accuracy of previous stages. The answers are categorized into two classes to arrive at the final accuracy. One type is the floating-point answers. Since the answer cannot be exact, we have allowed an error window of 5%. Values that are within this 5% range will be accepted. The other type are the String answers. Here, we consider an Exact Match metric. Answers that exactly match the ground truth values are considered towards accuracy.

We have compared our results with the PlotQA model. The PlotQA model was tested on 5860 questions from 160 images to obtain human accuracy. The paper reports the Human accuracy as 80.47%. On the DVQA dataset, the model has an accuracy of 58% and on the PlotQA dataset, the model has an accuracy of 22.52%. We have tested our model for a different number of images (5, 25 and 50) and the results are shown in table 9.1. The questions include open-ended, Yes/No, and structural.

Plot Type	Number of Images Tested	Total Number of Questions	Number of Correct Answers	Average Accuracy (in %)
Dot	2000	53970	25104	**46.965499**
Vertical	2000	47940	19898	**41.474200**
Horizontal	2000	49241	20128	**40.990114**
Line	2000	35353	14077	**36.669402**

Table 9.1 Accuracy of Table QA model for different number of Images

CHAPTER 10

CONCLUSION AND FUTURE WORK

We have proposed an alternative solution to perform question answering on statistical charts. The input chart is passed through the PED stage to obtain the bounding box predictions of the chart. It is then passed through the OCR stage to obtain the captured text from the image. Further, the output of the OCR stage is passed through the Semi-structure table generation to obtain a table in the form of CSV. Finally, the input question and the CSV file generated are passed through the Table Question Answering stage that outputs the predicted answer. TAPAS and TABFACT models are used for the purpose of Question Answering. The PED model produced an Average Precision of 87.014 % after training for 2 lakh iterations. Further, the Table QA model was tested for a different number of images and produced significantly better results.

Our Question Answering model is restricted to answer a limited number of questions. This limitation can be addressed in future work. The accuracy obtained by the PlotQA model and our model tested on TAPAS are significantly lower than the human performance. Hence, there is wide scope of improvement in the QA model and further research in this field.

REFERENCES / BIBLIOGRAPHY

[1] Kim, Dae Hyun, Enamul Hoque, and Maneesh Agrawala. "Answering questions about charts and generating visual explanations." Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems. 2020.

[2] Reddy, Revanth, et al. "Figurenet: A deep learning model for question-answering on scientific plots." *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019.

[3] Sharma, Monika, et al. "ChartNet: Visual reasoning over statistical charts using MAC-Networks." 2019 International Joint Conference on Neural Networks (IJCNN). IEEE, 2019.

[4] Methani, Nitesh, et al. "Plotqa: Reasoning over scientific plots." Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2020.

[5] Wu, Yuxin, et al. "Detectron2. 2019." URL https://github. com/facebookresearch/detectron2 2.3 (2019).

[6] Smith, Ray. "An overview of the Tesseract OCR engine." Ninth international conference on document analysis and recognition (ICDAR 2007). Vol. 2. IEEE, 2007.

[7] Herzig, Jonathan, et al. "TaPas: Weakly supervised table parsing via pre-training." arXiv preprint arXiv:2004.02349 (2020).

[8] Abadi, Martín, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." arXiv preprint arXiv:1603.04467 (2016).

[9] Kahou, Samira Ebrahimi, et al. "Figureqa: An annotated figure dataset for visual reasoning." arXiv preprint arXiv:1710.07300 (2017).

[10] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global vectors for word representation." Proceedings of the 2014 conference on empirical methods in natural

language processing (EMNLP). 2014.

[11] Kafle, Kushal, et al. "Dvqa: Understanding data visualizations via question answering." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[12] Berant, Jonathan, et al. "Semantic parsing on freebase from question-answer pairs." Proceedings of the 2013 conference on empirical methods in natural language processing. 2013.

APPENDIX A DEFINITIONS, ACRONYMS AND ABBREVIATIONS

Acronyms	Description
VQA	Visual Question Answering
ML	Machine Learning
NN	Neural network
CNN	Convolutional Neural network
LSTM	Long Short Term Memory
MAC	Memory Attention Composition
OCR	Optical Character Recognition
NLP	Natural Language Processing
TAPAS	TAble PArSing
OCR	Optical Character Recognition
PED	Plot Element Detection
RNN	Recurrent Neural Network
GloVe	Global Vectors for word representation
CSV	Comma Separated Values
JSON	JavaScript Object Notation

“VISUAL QUESTION ANSWERING ON STATISTICAL PLOTS”

by Sneha Jayaraman

Submission date: 20-Oct-2021 09:20AM (UTC+0530)
Submission ID: 1678760965
File name: Report.pdf (2.63M)
Word count: 14183
Character count: 84228

Dissertation on

"VISUAL QUESTION ANSWERING ON STATISTICAL PLOTS"

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology
in
Computer Science & Engineering

UE18CS390B – Capstone Project Phase - 2

Submitted by:

Sneha Jayaraman	**PES1201802825**
Sooryanath I T	**PES1201802827**
Himanshu Jain	**PES1201802828**

Under the guidance of

Dr.Mamatha H.R
Professor
PES University

June - December 2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

'Visual Question Answering On Statistical Plots'

is a bonafide work carried out by

Sneha Jayaraman	**PES1201802825**
Sooryanath I T	**PES1201802827**
Himanshu Jain	**PES1201802828**

in partial fulfilment for the completion of seventh semester Capstone Project Phase - 2 (UE18CS390B) in the Program of Study - Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period June - December 2021. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 7th semester academic requirements in respect of project work.

Signature **Dr.Mamatha H.R** Designation	Signature Dr. Shylaja S S Chairperson	Signature Dr. B K Keshavan Dean of Faculty

External Viva

Name of the Examiners	**Signature with Date**
1. _____	_____
2. _____	_____

DECLARATION

We hereby declare that the Capstone Project Phase - 2 entitled **"VISUAL QUESTION ANSWERING ON STATISTICAL PLOTS"** has been carried out by us under the guidance of Dr.Mamatha H.R , Professor and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester June - December 2021. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

PES1201802825	**Sneha Jayaraman**	
PES1201802827	**Sooryanath I T**	
PES1201802828	**Himanshu Jain**	

ACKNOWLEDGEMENT

I would like to express my gratitude to Dr.Mamatha H.R , Department of Computer Science and Engineering, PES University, for her continuous guidance, assistance, and encouragement throughout the development of this UE18CS390B - Capstone Project Phase – 2.

I am grateful to the project coordinator, Prof. Silviya Nancy J, for organizing, managing, and helping with the entire process.

I take this opportunity to thank Dr. Shylaja S S, Chairperson, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department. I would like to thank Dr. B.K. Keshavan, Dean of Faculty, PES University for his help.

I am deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr. Suryaprasad J, Vice-Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way. Finally, this project could not have been completed without the continual support and encouragement I have received from my family and friends.

ABSTRACT

Question answering systems have been used in various domains and applications like dialog systems, and medical domains for interaction with patients' therapy reports , scans and Xrays. In this work ,we have encircled the domain of data analysis involving statistical plots where question answering systems can be used. The statistical charts are used on a regular basis for data visualization to interpret the data and derive meaningful inference from them. This process can be automated using question answering systems. Users can impose a question to the system, for a particular statistical chart within the scope handled , the system must then process the query along with its pointers to/from the image data and provide the answer in the most accurate manner. Building such a system requires the usage of right architecture, right frameworks/tools and huge amounts of data.This work involves the research around existing visual question answering systems for graph-plots and aims to provide alternatives accounting to a novel approach . Once the model is built that satisfies the requirements, it can be deployed as a web application where a user can upload an image and, input a question, to propose the expected answer.

Visual question answering system in general, generates an answer to queries posed in natural language with the help of information extracted from the input image. Incorporating the ability to answer the questions on statistical charts is the aim of this research. In the development of our model, questions must be related to answers from fixed vocabulary or answers that can be extracted from the bounding box representation of an image or answers that can be queried from a structured table generated using visual elements. Plots related to bar plot, line plot, and dot plots and their variants have been accommodated/considered to be within the scope.

TABLE OF CONTENTS

10

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Statistical charts are an intuitive and simple way to represent data. They form a way of representing structured data in the form of graphical visualisations. Such graphical visualizations aid people in better interpreting features of data.Object detection in deep learning is a field that focuses on extricating localized datum from binary or color images. Therefore, it is useful to build a model that can localize and pick up visual data in statistical plots. It is one step towards the improvement in localized object detection capabilities.

Visual plots are commonly found in research papers, scientific journals, business records e.t.c. Therefore, automation of plot analysis through the means of question-answering aids an individual to draw statistical inferences quickly from them.

The most important benefit is that visual question answering models on charts will help data analysts question and understand plots on a large scale, and automate the decision-making capabilities in several sectors such as the financial sector.

Given this motivation, the aim of the project is to build a Visual Question Answering system which accepts statistical plots along with questions on the plot with respect to the elements of the plot (such as intersection of the curves, area under the curve, median value and few other varieties of such relational queries) and provides answers to the questions posed.

The system should discover relationships between elements of a plot and provide relational reasoning to answer questions on the plot.

Given an image of a statistical plot and a corresponding question, the model must be able to generate a representation of the image, parse it into an intermediary that is well interfaced with the workline , understand the query, and generate a suitable reply.

Therefore, it involves an understanding of localized image-element and the query language to be able to provide for visual reasoning. This work however restricts its scope to a certain amount of selective plots and their inner variants that are frequently occurring in most common data representations.Plots related to bar plot, line plot, and dot plots and their variants have been accommodated/considered to be within the scope.

CHAPTER 2

PROBLEM STATEMENT

Statistical plots are used widely by academicians and business employees because they are a simple way to represent data. They can be easily analysed and interpreted.

What if we could build an automated system that can analyse, discover relationships between elements of a plot and provide for relational reasoning capabilities or simply answer the queries posed on them? Such a system would mark a step towards machine reasoning capabilities.

With this motivation, the project aims to build a Visual Question Answering system that accepts statistical plots along with plot-specific questions concerning the elements of the underlying plot, such as the data-retrieval , mean , median ,range , min-max , difference , comparisons and few other varieties of such relational queries, to provide answers.

The difficulty with statistical plots is that even though they are images, they contain both structured and unstructured data. In the case of natural images, there are just visual elements to handle. However, that is not the case with statistical plots, since they contain both visual elements in the form of bars/sectors and textual elements in the form of axis labels and ticks.In short they contain multiple objects within an image where the object is a plot element and the image is the graph itself. To add to this, the size of objects that are there in natural images are constrained to small, medium and large in general. In the case of statistical plots, the aspect ratio is much more varied. For example, in the case of bar plots, there could be bars that are extremely small, and bars that are extremely large on the same plot.

The measure of the accuracy of prediction in the case of images is normally IOU (intersection over union). The success criteria for a correct prediction in the case of natural images is normally 50 per cent. The same rate is insufficient for statistical plots. This is because we want the prediction for a bar value (in the case of bar plots) to correspond to the

actual value as seen on the graph, as close as possible. Therefore the adequacy criteria for a successful prediction is much higher.

The above-mentioned factors highlight the differences in natural images and statistical plots. Therefore, state-of-the-art object detection models do not suffice for this application. The aim here is to apply deep learning concepts to come to an acceptable solution to the problem of analysing statistical plots using machines. In this work we make use of object detection algorithms rather than image detection algorithms such as faster R-CNNs with feature extractors which are localized , FPN , deep nets like Resnet 101 , 50 or Resnext to constitute an object detection model followed by a set of utility models to perform character recognition concluding with a question answering module.

CHAPTER 3

LITERATURE SURVEY

In the following subdivision , we present the current understanding and knowledge of the area along with reviewing substantial findings that help shape, inform and reform our study leading us to set the platform to further envisage the possible improvements that can be brought up.

3.1 Background on Statistical Charts modelling for QA system

This section briefs the papers consulted and thoroughly reviewed to gain information on background,data used, the architecture style , the patterns and the proposed/existing methodologies being used in the domain of question answering system for statistical charts.

3.1.1 Answering Questions about Charts and Generating Visual Explanations [1]

Summary

The paper under consideration proposes the chart question answering system that generates chart specific answers along with the explanation on how the answer was obtained. The visual attributes of the charts are transformed into references to the data. State-of-the-art ML algorithms are used to generate answers and its corresponding explanation.

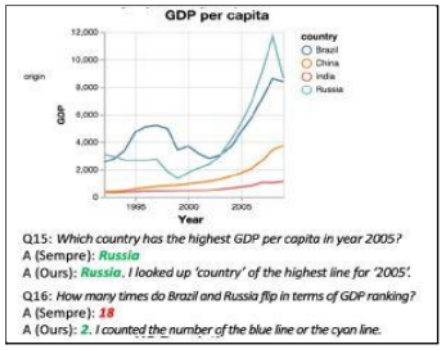


Figure 3.1

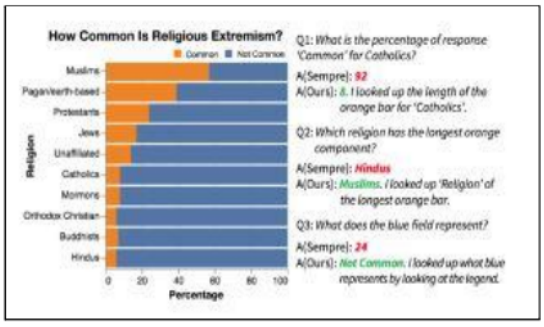


Figure 3.2

Figure 3.1 illustrates two question and answer pairs for a line plot. The results are compared between the Sempre model and the model proposed.

Figure 3.2 illustrates three sample question and answer pairs for a stacked horizontal bar plot.

Dataset

The compilation of dataset consists of 52 charts, congregated from four contrasting sources:

- The Vega-Lite Example Gallery
- Graphical Charts in Pew Research Reports
- D3 charts that are accumulated from the internet
- Charts fabricated from tables present in the WikiTableQuestions data compilation.

The questions, answers and their explanations were manually generated.

Dataset Counts :

In union, the compiled data includes 5 line charts and 47 bar charts (32 simple, 8 grouped, 7 stacked).In total, 52 charts are synthesized that includes 629 questions, 866 answers and 748 explanations for the answers generated , this data is considered for our work as well.

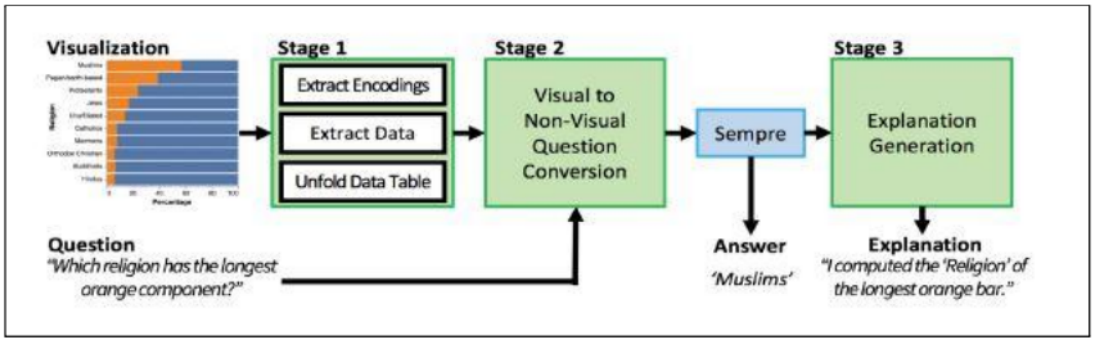
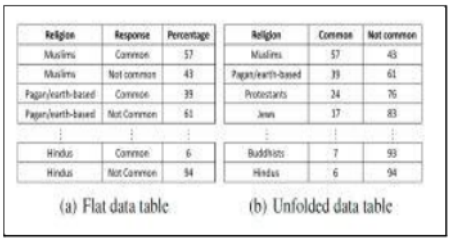


Figure 3.3



Religion	Response	Percentage
Muslims	Common	57
Muslims	Not common	43
Pagan/earth-based	Common	39
Pagan/earth-based	Not Common	61
⋮	⋮	⋮
Hindus	Common	6
Hindus	Not Common	94

(a) Flat data table

Religion	Common	Not common
Muslims	57	43
Pagan/earth-based	39	61
Protestants	24	76
Jews	17	83
⋮	⋮	⋮
Buddhists	7	93
Hindus	6	94

(b) Unfolded data table

Figure 3.4

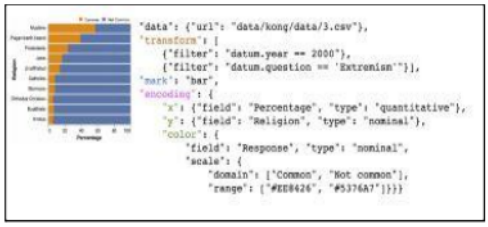


Figure 3.5

Figure 3.3 illustrates the pipeline of this model for the question answering system ,***Figure 3.4*** shows the dataset format and ***Figure 3.5*** shows the unfolded data table.

Methodology Proposed

Firstly, visual encodings like the pinnacle aka height of the bar, color/shade grading of the line, etc. are extracted from the charts. The input question is transformed, replacing any visual references made by chart elements to the non-visual references to the data fields and data values. The Unfolded table is passed through Sempre (QA algorithm that functions with relational data tables instead of any statistical charts) to generate the answer. Sempre model converts the input question into a lambda expression. It then performs query execution on the data table generated to produce the answer.The lambda expression obtained is transformed to visual explanation for the answers by a method known as template-based translation.

Formal Steps Stated in the paper

- **Stage 1**: Extract Data Table and Encodings
- **Stage 2**: Visual to Non-Visual Question Conversion
    - Step 1: Mark detection
    - Step 2: Dependency parsing

- Step 3: Visual attribute detection
- Step 4: Visual operation detection
- Step 5: Apply encodings
- Step 6: Natural language conversion

- **Stage 3**: Explanation Generation
  - Step 1: Natural language conversion
  - Step 2: Implicit field recovery
  - Step 3: Redundancy Cleanup
  - Step 4: Sentence Completion
  - Step 5: Encoding application

Merits

The paper not only provides accurate answers to the questions but also provides an explanation on how the answer was obtained. The model produces valid answers and their corresponding explanations as opposed to the Sempre model that could not answer the questions correctly.

Demerits

The system cannot handle certain types of questions that involve synonyms of the features present in the chart. There is scope for improving the explanation provided for the answers.

3.1.2 FigureNet: A Deep Learning model for Question Answering On Scientific Plots [2]

Summary

This model uses a CNN with depth-wise convolutions, LSTM and feed-forward NN to handle the task of answering questions on plot such as pie and bar on a dataset that's named FigureQA.

Dataset

The dataset used is the FigureQA dataset that contains more than a million questions with answers on various types of scientific plots. This dataset has plots with elements that are color-coded. There are a total of 100 colors that are used across both training and test datasets. Therefore, it is possible to distinguish between elements without the need of character recognition for text. Additionally, it provides pre-annotated data with bounding boxes.

Template
Is X the minimum?
Is X the maximum?
Is X the low median?
Is X the high median?
Is X less than Y?
Is X greater than Y?

Figure 3.6

Figure 3.6 shows the template of questions in the FigureQA dataset.

Model

The FigureNet architecture, as proposed, can handle the task of answering relational questions on pie charts and bar plots. It uses the FigureQA dataset, that consists of statistical plots with plot elements that are color coded. Additionally, it is guaranteed that the plot consists of no more than 11 plot elements, and that there are 100 different colors that are used to represent plot elements. The end goal of the FigureNet model is to be able to answer questions in a binary yes/no manner. To be able to do this, the authors have divided the task into subtasks as follows.

- Spectral Segregator Module - Identify plot elements and color of the plot elements.
- Order Extraction Module - Identify and quantify the values associated with each plot element, and then sort it into increasing order.
- Question Encoding - Provide an encoding for the question.
- Question Color encoding - Identify mentions of color in the question.

Methodology proposed

Spectral Segregator Module:

23

This module is used to identify individual elements and color of these elements of the plot. 128 x 128 x 3 image is passed as input to a CNN that uses depth-wise convolutions to identify colors and separate channel information. This way we don't just get an aggregate map of the image. The output here is a 512 dimensional image representation. This image representation is passed to a 2-layer LSTM to get the most probable color for each element. There can be at most 11 elements in a plot.

Order Extraction Module:

This module is used to identify and quantify the statistical values of each plot element and their relative order. It is similar to that of the previous module except that now the output of the LSTM will be the ordering for each of the plot elements starting from 1.

Question Encoding and Question Color encoding:

This module uses 2 layers of LSTM cells (many to one model) to produce a question encoding.

Final feed-forward NN:

All the four modules are concatenated and passed onto a feed forward NN to produce a binary (Yes/No) answer using the Sigmoid Activation function for the output layer.

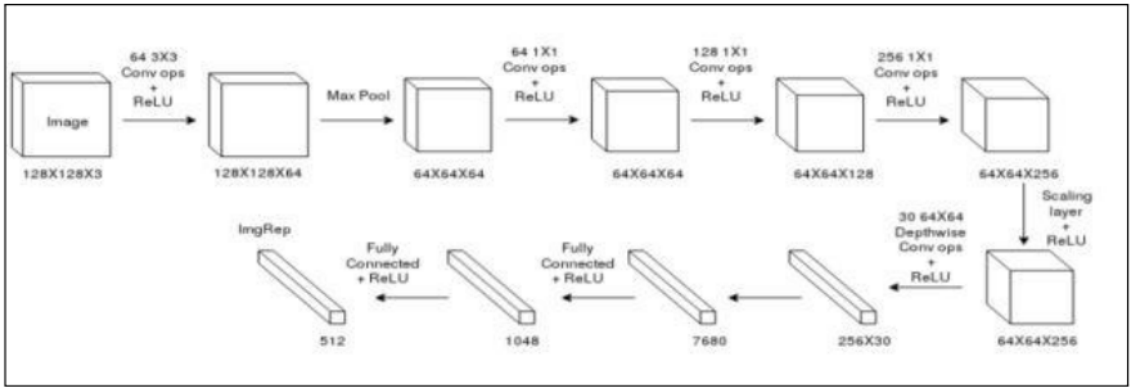


Figure 3.7

```
[Royal Blue, Aqua, Midnight Blue, Purple, Tomato,
STOP, STOP, STOP,
STOP, STOP, STOP]
```

Figure 3.9

Figure 3.7 shows the architecture of the Spectral Segregator Module that uses layers of convolution and max pool, followed by depthwise convolutions and feed forward layers.

Figure 3.8 shows the rest of the spectral segrator module that uses a custom LSTM architecture. The input here is the image representation that is obtained as the output from the architecture in Figure 3.7.

Figure 3.9 shows a sample output as obtained from the architecture in Figure 3.8.

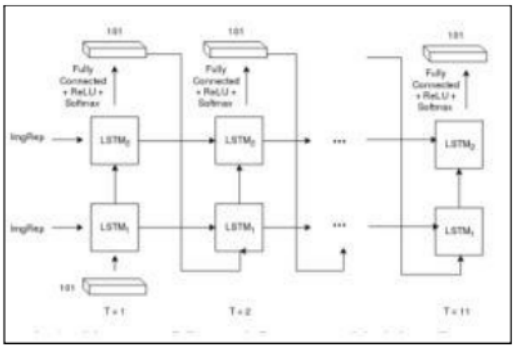


Figure 3.8

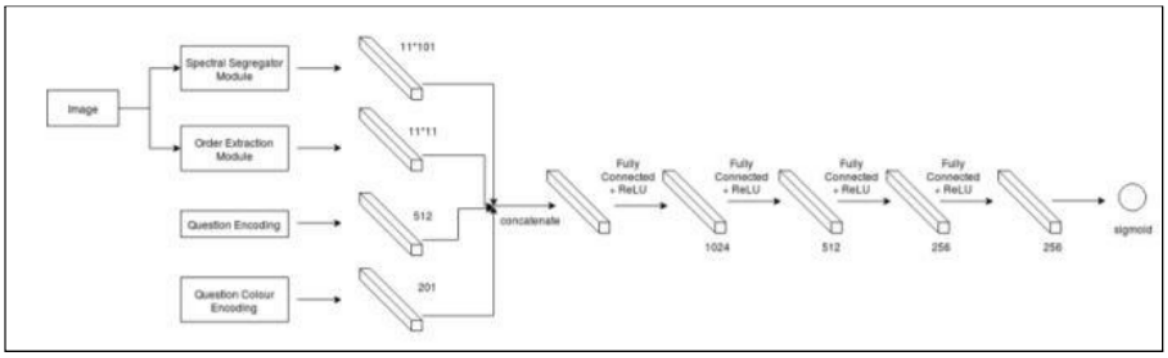


Figure 3.10

Figure 3.10 shows the final feed forward architecture.

Figure Type	CNN + LSTM	RN(Baseline)	Our Model	Human
Vertical Bar	60.84	77.53	**87.09**	95.90
Horizontal Bar	61.06	75.76	**82.19**	96.03
Pie Chart	57.91	78.71	**83.69**	88.26

Figure 3.11

Figure 3.11 depicts a table comparing accuracy values for each type of plot.

Merits

The model performs significantly better than the baseline models. This is because the architecture doesn't use the traditional CNN, instead uses depthwise convolutions. Additionally, the model used lesser training time as articulated in the paper.

Demerits

The model works on only bar plots and pie charts. It is capable of only binary reasoning, and not capable of answering open-ended questions. It makes use of the FigureQA dataset, thereby making use of the property of the charts being color coded.

3.1.3 ChartNet: Visual Reasoning over Statistical Charts using MAC-Networks [3]

Summary

Proposed paper solves the problem of reasoning over charts (only bar and pie charts) using MAC-Network (Memory, Attention, and Composition). The model is capable of answering open-ended questions and gives chart-specific answers. The classification layer of MAC is substituted by the regression layer and constructs a boundary for the text that corresponds to the answer. OCR is used to read the text and display the answer.

Dataset

The data synthesized by the model consists of bar-charts and pie-charts. The bar charts dataset consists of vertical bars and was created by varying the height and number of colors of bars. Data for pie-charts is created by varying the colors of sectors and also the angles between them. The annotations of the bounding box that are present over the chart images are saved to give chart specific answers. In total, 20k, 5k and 5k image question pairs for training, validation and testing are created, for both bar charts and pie charts.

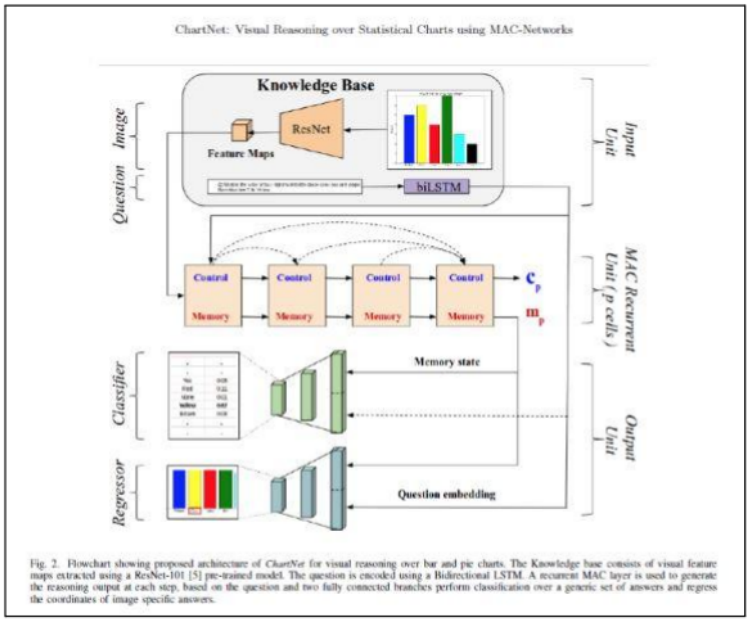


Fig. 2. Flowchart showing proposed architecture of *ChartNet* for visual reasoning over bar and pie charts. The Knowledge base consists of visual feature maps extracted using a ResNet-101 [5] pre-trained model. The question is encoded using a Bidirectional LSTM. A recurrent MAC layer is used to generate the reasoning output at each step, based on the question and two fully connected branches perform classification over a generic set of answers and regress the coordinates of image specific answers.

Figure 3.12

Figure 3.12 shows the architecture of the model proposed.

Methodology proposed

ChartNet network consists of three layers: Input unit, MAC Cell, Output unit.

Input Unit

Bar or pie chart is given as an input and corresponding question. Features from the images are extracted using ResNet101 deep CNN architecture. Knowledge base is defined to depict the height and the width image. Questions are converted into word embeddings and they are further processed using the biLSTM model.

MAC Cell

It represents a recurrent unit which consists of three components : Control, Read and Write. It is defined to reason the questions posed and also to implement them.

Output Unit

This unit consists of two networks : Classifier and Regressor. Classifier network predicts the probability distribution over all of the predefined answers by using softmax normalization. The regressor network is used to provide chart-specific answers.

Merits

Automated method for question answering over open-ended questions. MAC-Network included with the regression layer helps the model make predictions over unseen answers.

Demerits

The model is not generic and works only for vertical bar charts and pie charts. Model cannot answer questions that require numerical operations.

3.1.4 PlotQA: Reasoning over Scientific Plots [4]

Summary

A step towards developing a holistic plot based visual question answering model , which can handle both in vocabulary and open ended queries using a hybrid approach.

Dataset

The graphical summaries are produced from data provenanced from the organizations like the World Bank, government maintained sites to name a few , thereby having a large vocabulary of graph parameters like ticks , and a wide variety of range in data instances. Out of vocabulary questions are generated and they are not straight forward as they are generated on the basis of 70 plus patterns extracted from 7,000 public flock questions asked by data collectors on a sampled set of 1000+ plots.

Datasets	#Plot types	#Plot images	#QA pairs	Vocabulary	Avg. question length	#Templates	#Unique answers	Open vocab.
PlotQA	3	224,377	28,952,641	Real-world axes variables and floating point numbers	43.54	74 (with paraphrasing)	5,701,618	Present

Figure 3.13

Answer (A) Type	Question (Q) Type		
	Structure	Data Retrieval	Reasoning
Yes/No	36.99%	5.19%	2.05%
Fixed vocabulary	63.01%	18.52%	15.92%
Open vocabulary	0.00%	76.29%	82.03%

Figure 3.14

Figure 3.13 summarises the dataset used. ***Figure 3.14*** shows the long range distribution of Query and Response types from the data compilation in the PlotQA data compilation.

Model

Existing Works :

Existing solutions for VQA fall under two categories: (i) extricate the response from the graphical data input (like in LoRRA) or (ii) respond with an answer to the query posed based on the existing vocabulary (like in SAN and BAN). Such approaches seem to work well for data compilations as portrayed in DVQA , but under fit for PlotQA with a considerable majority of out of vocabulary queries.

Plot QA's Model :

This is a composite model encompassing the below features and entities:

(i) a binary categorizer for decision making as to whether the input query be responded with from an existing vocabulary or demands an advanced treatment.

(ii) a simpler question categorizer to respond to queries of the simpler or complex treatment type.

(iii) The process pipeline with the CNN and LSTM combination setup.

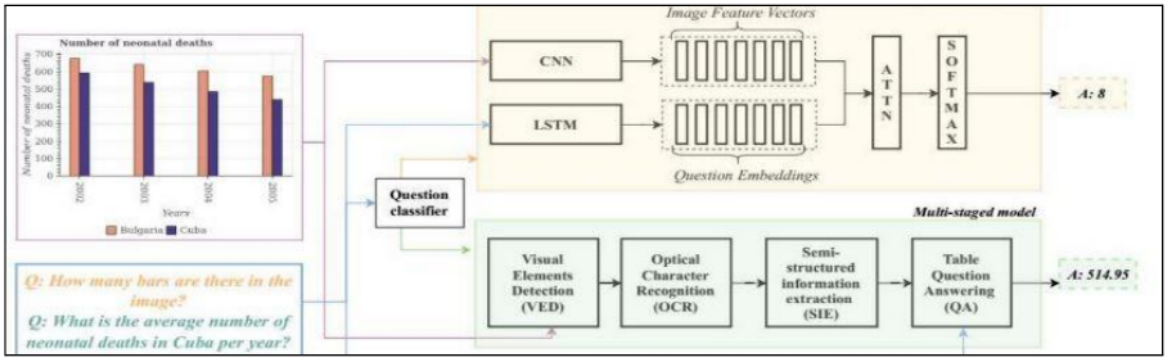


Figure 3.15

13

Figure 3.15 shows the architecture of the model proposed. Observe the two pipelines.

Methodology proposed

This is a composite mix match model encompassing the the detailed entities as follows: (i) a binary classifier for categorizing the complexity of the question and to provide decision about if it can be handled by in vocabulary or does it need assistance of out of vocabulary processing pipeline (ii) a simpler question categorizer model to respond to queries of types as mentioned in (i),finally (iii) presence of a multi-staged model encompassing four software modules as briefed in the following section to deal with the lower half of the pipeline which is the out of vocabulary questions.

1. Visual Elements Detection Module

Primary task to perform is to extricate the visual entities by demarcating / annotating bounding boxes around those entities and categorizing them into the appropriate groups.

Upon comparing all methods, it is observed clearly that the Faster R-CNN model in union /combination with Feature Pyramid Network (FPN) outperforms the existing architecture combinations and hence that becomes an apt fit for the visual element detection module.

2. Object Character Recognition Module:

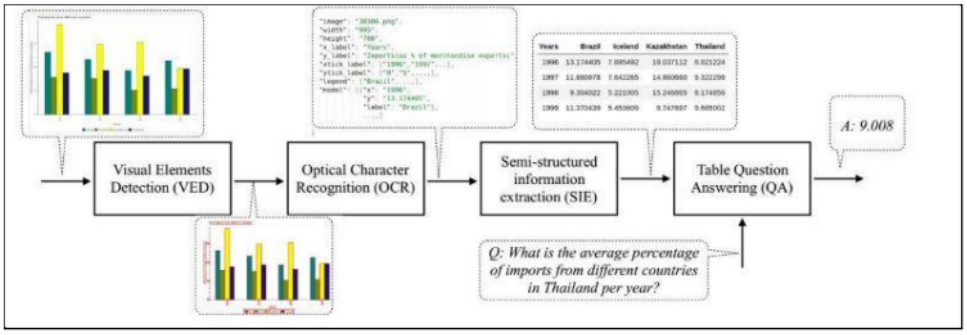
The common visual entities of the graphical summaries like legends, tick labels to name a few . accommodate numerical and textual data. For the purpose of extricating this data from bounding boxes annotations , the avant-garde OCR model is used.

3. Semi-Structured Information Extraction Module:

The penultimate phase of the pipeline. The data captured as results in the form of json/dictionary from the previous phase is formatted into a table structure using this module.

4. Table Question Answering Phase:

The ultimate final phase of the processing pipeline is to answer queries by superimposing them on the semi-structured pivoted table version of the image which has now been dissected. This is akin to answering questions from the WikiTableQuestions dataset , so the similar process is recreated to obtain results.



26

Figure 3.16

Figure 3.16 shows the proposed multi-staged modular/unit-wise pipeline.

Merits

Can handle out of the vocabulary questions (OOV) along with in-vocabulary question types. The data collected to prepare graphs in the dataset are from various financial and business resources. Blurs the line of difference between computerized-data plot datasets and real life data summarized in graphs and query patterns.

Demerits

The model is not generic and works only for bar charts , line charts and dot plots. There exists a want/development in regard to more precise visual element detecting (VED) modules to enhance responses over the queries posed on the plots.

CHAPTER 4

DATA

The following chapter describes the data used to build the question answering system for statistical plots. Building the right questionnaire with right visualization is critical to build a high accuracy model.

4.1 Overview

Statistical plots are used to represent the data and help in deriving insights from them. Some of the basic charts are bar charts, dot plots and line charts. Visual aid is required to analyze these charts and extricate the objects/plot members from them and answer some of the questions related to them.

4.2 Data Format

Statistical plots are used to learn about data and their important features. Building a visual question answering system requires statistical plots and well-defined questions and answers. The detailed description of the charts and the question answer pair is provided in further sections. The over data compilation comprises of graph images which are supported by a set of annotation files that are well formatted using a json structure.The images and their annotations must be correlated/combined to get the bounding boxes around the training images , that is the prep step before the image+annotated data is passed to training for object detection.

4.3 Statistical Charts

Statistical charts from one of the inputs to the visual question answering system. The types of charts that we have constrained to are bar charts, line plots and dot/dot line plots .

A typical plot in general irrespective of its type will include

- Title of the plot/graph
- Xlabel and Ylabel
- Legend data if any
- Xticks and Yticks

There are different types / variants of bar charts like vertical charts, horizontal charts, simple, and group charts.Whereas dot plots and line plots have 0 or no variants.

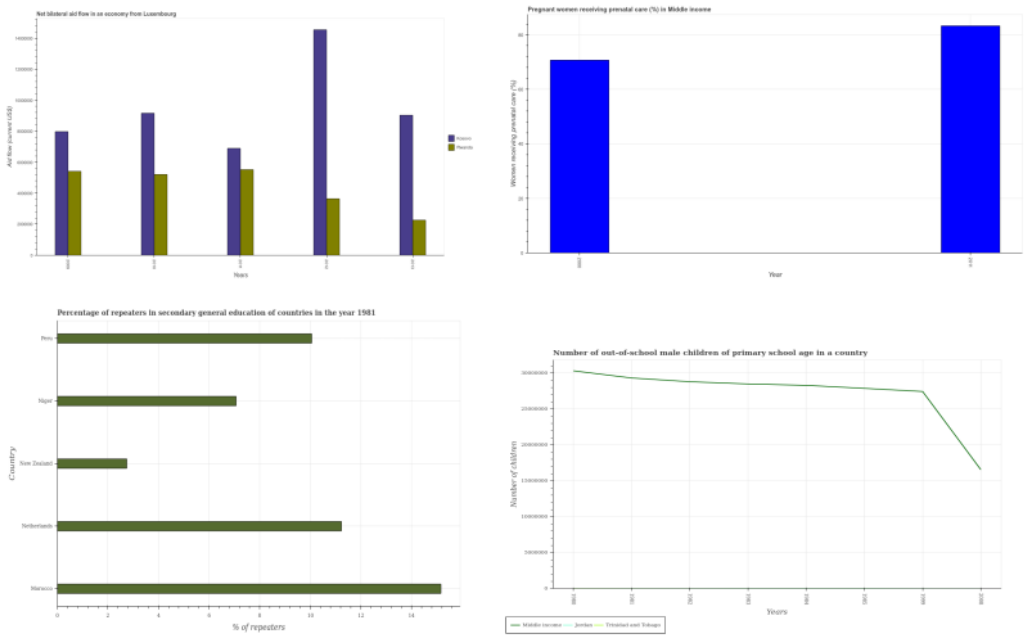


Figure 4.1 (Image Courtesy:PlotQA: Reasoning over Scientific Plots)

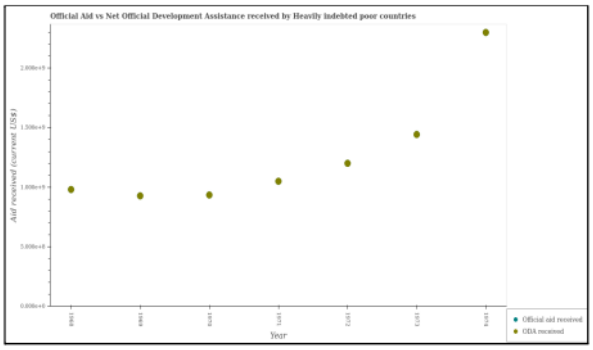
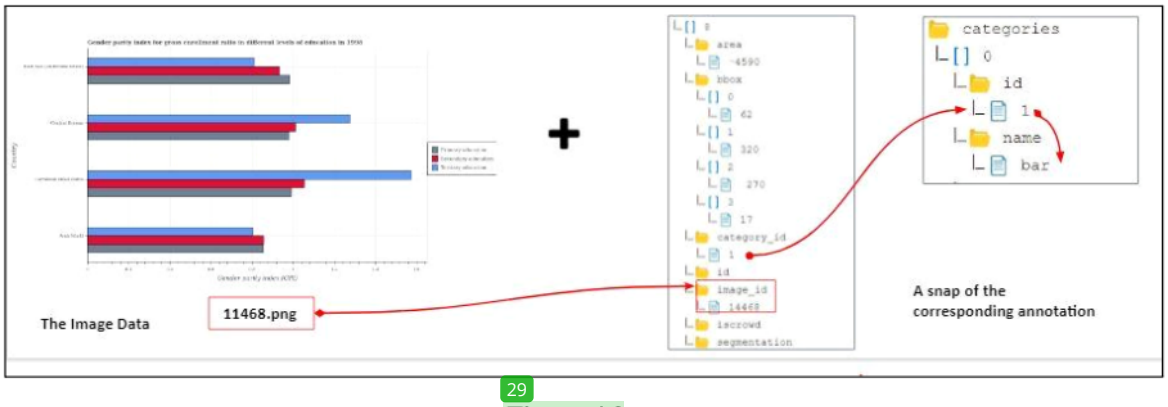


Figure 4.2 (Image Courtesy:PlotQA: Reasoning over Scientific Plots)

Figure 4.1 represents [top-left to bottom-right] vertical grouped bar , simple vertical bar , simple horizontal bar and simple line plot. ***Figure 4.2*** represents a dotplot.

The plot element detection stage will intake an excess of these images with their corresponding annotations , the annotations of the images act as a catalog for each image , they have information regarding the bounding box points of reference for every significant plot element present in a graphical image . With the help of this data , our object detection model tends to learn the possible bounding boxes location within any inference image passed on to it.



29

Figure 4.3

Figure 4.3 shows the raw image data and the corresponding annotation data which acts as a catalog for the image . It is to be noted that both of them exist separately.Necessary preprocessing is done to combine them together , so that the plot elements have their bounding box points around them before training.

4.4 Question And Answers

Every image has a list of questions and their answers stored in the repository. Descriptive questions or Open-ended questions are generated for each image. The answers for them are also stored. Some of the examples of the question answers are shown in the figure. During the training of the model, the chart list of questions are passed as the input.However the question input is only used in the final stage wherein it is fed into a weak supervised table question answering model. The model for table question answering is already pre-trained in case of this work to predict the answer. During testing, only the chart and a question is taken as the input, and the model predicts the answer.

At a lower level , firstly the image is passed onto a trained model and inference is performed to get the bounding box information of all the involved plot elements , next the points of references to the graph elements is passed along with the image to the OCR stage , in which the localized text content is extricated , after which we have utility modules to map the complete image data into its tabular equivalent and generate the csv , now things become simpler because the plot data is no more an image but it is a semi-structured table on which table question answering can be performed , just as in case of the wikitables questions dataset. The questions that are handled in our work are related to data-retrieval following a simple select, project or select cum project vice-versa , statistical mean , min - max , boolean truths , range based queries , simple summations ,summations across an interval from the graph data , trend wise queries , differences with few constraints and restrictions laid on them.

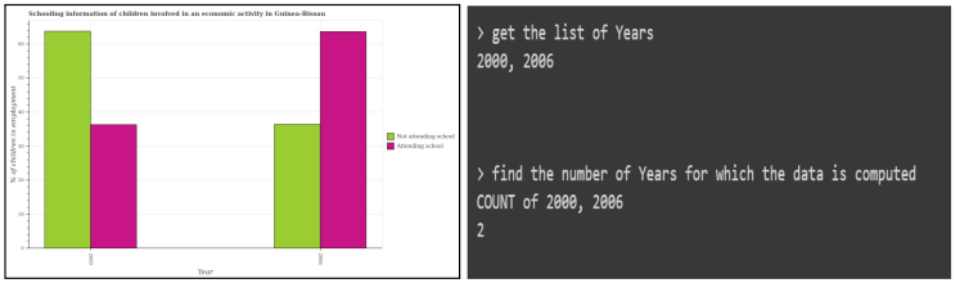


Figure 4.4 indicates a graph {grouped vertical bar} on the left and the question posed on it

with the predicted answer on the right.

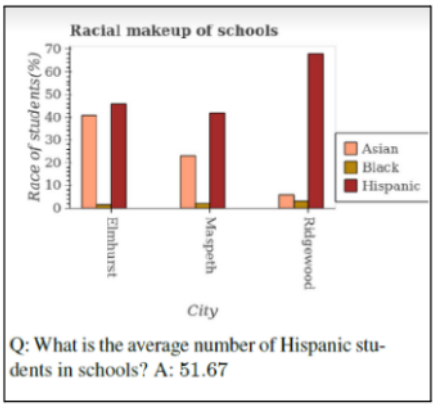


Figure 4.5

Figure 4.5 is a sample of a query associated with the graph concerning a numerical/arithmetic operation of averaging a particular category.

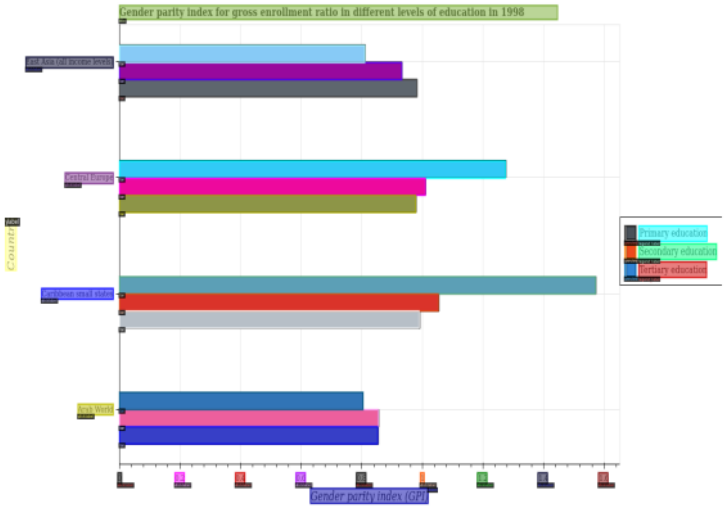


Figure 4.6 The output of an image in Figure 4.3 superimposed with its annotation (pre training) . Those are the bounding boxes which were generated manually

CHAPTER 5

SYSTEM REQUIREMENTS SPECIFICATION

5.1 Project Scope

Goal

The system should be able to answer questions on Vertical and Horizontal Bar Graphs {simple , grouped}, dot line Charts, dot plots and Line Plots and questions within the scope of handling . Provide an user interface through which the end user can upload the image and a corresponding question.

Limitations

The system will not be able to answer questions on the smoothness or the roughness of the plots. It can only answer relational queries – queries with respect to the other elements of the plot. Handles only graphs considered within the scope of work and there do exist some constraints on the questions posed on the graphs .The questions that are handled in our work are related to data-retrieval following a simple select, project or select cum project vice-versa , statistical mean , min - max , boolean truths , range based queries , simple summations ,summations across an interval from the graph data , trend wise queries which have certain keywords required to be present in the queries.

5.2. Product Perspective

Visual question answering specific to statistical plots, has less prevalent work done. It is one step towards the improvement of machine reasoning and pattern identifying capabilities and object detections in scientific plots.

5.2.1 Product Features

1. Input: The input to our model is an image of a statistical plot covered within the scope of work and a corresponding question on the plot.

2. Model: Our model will take in the input, process the visual image using the object detection model further converting the objects obtained into a consolidated semi-structured format and parse the query, concatenate the inferences from both and produce an output. Thus, it combines the technicalities of object detection capabilities and query language understanding.

3. Output: The output of the model is the answer to the question.

5.2.2 Operating Environment

The model will be made available as a web application; hence it does not depend on the underlying Operating system and its versions. It only requires browser support of HTML5 and above with a flask oriented backend to serve the requests consisting of input image and supporting query along. Use of the model can be done via the internet. Colab Pro offers a linux environment , however the os utilities are used only for the backend functionality , none of that affects the user interface.

On the server side, the model can be saved and loaded when necessary(post training , the weights are saved to perform inference and generate further outputs for intermediate stages). Therefore, the platform must be available during requests , must provide sufficient disk space of minimum 15GB for hassle free loading of dataset and saving intermediate results , RAM of minimum 12GB for smooth processing without hiccups, GPU support of 16GB to get the deep nets trained over a period of time for longer iterations . For the training and testing phase a linux based environment is needed and it has been done so.A linux backend plays a major role in serving the request , but the technicalities are hidden and unrelated to the end user.

5.2.3 General Constraints, Assumptions and Dependencies

The project focuses on model building and providing for accurate and reliable answers to the questions posed on the statistical charts. Hence, there is less focus on the security considerations. However, our solution will consist of an interface to the model that facilitates image upload and questions on the image. Most of the assumptions , limitations and dependencies are covered in the section regarding the limitations. The system produces fair results only on the graphs and queries handled well within the scope.

5.2.4 Risks

Operational risk in terms of management and support for the product is a possible risk case.

Functional Requirements

Question Answering System for Charts take in statistical charts and questions related to them as input. Visual features from the charts have to be extracted and preprocessed. This can be done using techniques like object detection processing and Optical Character Recognition (OCR). The questions provided as an input need to be of the type handled by the scope of this model. Important details from the image and the questions need to be mapped (done here through OCR and semi-structured table creation) and the corresponding answer should be predicted (table question answering). Deep learning methods and architectures will be employed to predict the output both at the image level and questionnaire levels. Inputs are validated by the system to recognize only statistical charts. Any other images will be responded with poor results as they aren't seen by the model previously or rather untrained.The system will be trained on huge amounts of data and the results will be validated against the true answers using suitable methods. The parameters for the object detection model will be tuned to provide the most bounding box capturing capability and a proper pre-trained table question answering module will have to execute the queries on the tabular data and respond with an answer to the end user.

5.3. External Interface Requirements

5.3.1 User Facing Interfaces (UI)

The user interface is a web application which will take a chart and user-specific question. Users will be allowed to upload an image from their local drive. A text box will be provided that will accept the questions. The model will take the images and questions as the input and run in the backend , where the low level operations will be performed to feed in the image and questions to the pipeline. It will return the most probable answer and display it on the screen. Additional data like the confidence of the answer can also be displayed. A trained model will be deployed on the web server, and hence, the output should be produced within a few seconds.

5.3.2 Hardware Requirements

Deep learning tasks are computer intensive. The hardware requirements to build and train the model will require a minimum of 12GB RAM.A disk space in excess of 10GB for hassle free operation and a GPU support in excess of 10GB. Optionally, the model can be trained on the cloud to avoid physical hardware limitations , in platforms that offer compute as service. Once the model is trained, the testing phase can be done on the same cloud commodity using the pay as you go method.

5.3.3 Software Requirements

Question Answering Systems are built using deep learning models and NLP techniques. Python (Version: 3.6 , 2.x or more), NLP libraries object detection toolkits such as detectron , open source deep net frameworks, image processing tools and basic os and python-ic utilities supported with intermediate helper modules. Web frameworks like React or Flask are required to build and deploy the web application. The system can be built on an OS with linux distribution. Versioning of the model will be done using GitHub or possibly a dedicated google account can

be provisioned to carry out the activities , because there is a high amount of storage required to maintain a repo . Separate notebooks can be created as per requirement with helpful bookmarks to make it hassle free for the users to run through them.

5.4. Non-Functional Requirements

5.4.1 Performance Requirement

- **Usability** : The trained model must be available at ease to use it ,just by choosing an image input from the local drive or file-system . Similar to drag and drop or attaching files. The user must be able to navigate through the interface even with minimal exposure towards computing technologies.

- **Reliability** : As the product is developed by following practices in deep-learning , machine learning and imaging , there is no certain fixed level of reliability that can be set for the product. Reliability is not completely independent of the inputs to the model , hence reliability varies with respect to the context and type of inputs passed in provided they comply with the assumptions and restrictions of the model.

- **Maintainability** : As the product is just a model , maintaining the model isn't a difficult task , it only requires a cloud machine , a drive to support storage on cloud so that it can persist and to reside on and a browser / interface to access.

- **Performance** : The model is expected to draw statistical inferences based on the question and the input passed with a good and acceptable accuracy level relative to ground truth or human eval.

- **Robustness** : The model must be robust enough to handle any of the graph images and questions related to its scope of operation. The model must yield good performance throughout

the pipeline for any input from the wide range of possible inputs pertaining to the scope of the model.

5.4.2 Safety Requirements

Safety requirements pertaining to this product are minimal as it isn't deployed onto a live physical environment . The decision process for users , such as passing image inputs or attaching them to gain access to model /product and data must follow a need-to-know principle, which states that access to internal data must be available only to the designers of the model and shouldn't be exposed to the end users.

5.4.3 Security Requirements

Data Sharing : As there are lots of graphical images to be collected , the data and statistics storage along with logger data will be done to maintain the correct functioning of the model and to reconstruct what went wrong in case of any system - failures by constructing checkpoints. The datasets if artificially generated need to be secured locally and not be made available for commercial usages.

Model security : The model trained needs to be protected on a local machine or on cloud storage if deployed onto the cloud . Modification of ownership to only allow viewable access can be enabled to outsiders , versioning of the model weights can also be done to build over the training and enhance the performance.

CHAPTER 6

DETAILED SYSTEM DESIGN

6.1 High Level Design:

Question Answering System for Charts take in statistical charts and questions related to them as input. Visual features from the charts have to be extracted and preprocessed. This can be done using techniques like image processing and Optical Character Recognition (OCR). The questions provided as an input need to be pre-processed using NLP techniques. Important details from the image and the questions need to be mapped and the corresponding answer should be predicted. Deep learning methods and architectures will be employed to predict the output. Inputs are validated by the system to recognize only statistical charts. Any other images will be rejected by the system or rather the model doesn't recognize the graph image. The system will be trained on huge amounts of data and the results will be validated against the true answers. The parameters for the model will be tuned to provide the most optimal answer as the output.

6.1.1 Current existing works

There have been attempts in the recent past to improve machine reasoning capabilities through visual question answering systems on graphical plots. The RNN architecture and the CNN-LSTM architecture form baseline comparison models with an accuracy of 75% and 60% respectively. This in comparison to human accuracy falls short by a large margin.A recent paper publication introduced the FigureNet architecture that was able to achieve an accuracy of approximately 85% on an open-source dataset. This however, only gives a yes/no binary output to a question posed, and is limited to only bar and pie charts. Another adaptation to this model, showed significant enhancements in terms of being able to answer open-ended questions on a different synthesised dataset. This domain of visual question answering on statistical plots, however, has a lot of scope of improvement in terms of future enhancements of these models. There is a possibility of expanding the types of charts to those beyond bar and pie charts or even improving on accuracy through model adaptation. In our work we made an attempt to finetune the existing work and also tried out the alternatives for table question answering stage.

6.1.2 High Level System Design

High Level Design Diagram:

Input: There are two inputs to our model that the user needs to provide. The first input is an image – that depicts a statistical plot and the second input is a relational question on the image.

Our Model: The design consists of 3 primary components.

1. **Image Encoding Module /Plot elements detection model**

This is a module that takes in the image as its input along with the annotations of the image , correlates them together.This module consists of a deep learning FASTER RCNN module for object localization , because our main aim is to localize and produce bounding boxes around the plot elements and extricate them rather than classifying an image .This module produces the bounding box annotation of all involved plot elements captured by the object detection model which is fine-tunes and trained by us. Bounding boxes are a vector representation of the plot elements in a graph , which refer to the coordinates of the object (top_x , top_y , bottom_x , bottom_y) ; we effectively need only these 4 points to draw a bounding box around a plot element . Bounding box values of the plot elements like xlabel , title ,ylable , the bar , the line , the dot are all obtained.While an image classification network can tell whether an image contains a certain object or not, it won't say where in the image the object is located. We are in the quest of **locating the plot elements** and **not just classifying the plot.** Hence we just don't need just a CNN , we need an object detection neural network like R-CNNs we need to decide over the **object detection model**.

- We chose **Faster-RCNN** as our object detection model which is a **descendant from the R-CNNs series**.
- The heuristic behind doing so can be inferred from the below observations and also due to the presence of the RPN (Regional proposal network is known for locating feature targets accurately) .

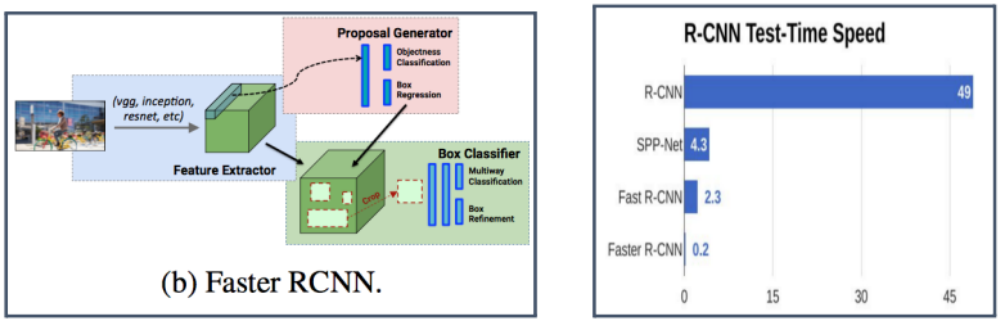


Figure 6.1 Indicating the Object detection model we chose

We also need to decide on the feature extractor model for serving as the backbone/feature-extractor for our faster-RCNN Setup.We Chose Resnet-101 to be our feature extractor due to skip connections and residual blocks which can reduce the problem of vanishing gradients in a very deep network like Resnet-101.

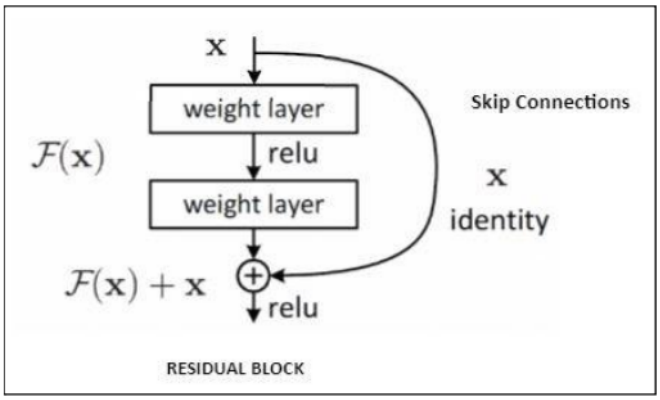


Figure 6.2 Indicating skip connections in a Resnet-101 model

2. **Question Encoding Module / table question answering stage**

The input to this module is the question (English Language) and the output is a question embedding. The question embedding captures all of the relevant information in the question in a format that is suitable for further modelling. This module needs tabular data on which the question answering can be performed . Hence the output of the image module must be further

structured using other utility stages so that a semi structured table is obtained , on which queries can be executed.

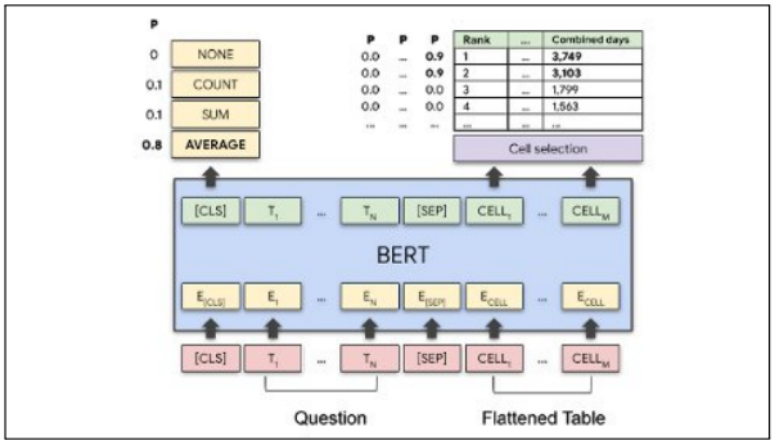


Figure 6.3 A bert based model - Tapas to perform table question answering

3. **Result Estimation Module**

The output of the image module would be a list of (element_type , confidence score ,bounding box vector) . The length of the list indicates the number of elements in the input plot . Now that bounding box annotations are obtained ,next the points of references to the graph elements is passed along with the image to the OCR stage , in which the localized text content is extricated , after which we have utility modules to map the complete image data into its tabular equivalent and generate the csv , now things become simpler because the plot data is no more an image but it is a semi-structured table on which table question answering can be performed , just as in case of the wikitables questions dataset. The questions that are handled in our work are related to data-retrieval following a simple select, project or select cum project vice-versa , statistical mean , min - max , boolean truths , range based queries , simple summations ,summations across an interval from the graph data , trend wise queries , differences with few constraints and restrictions laid on them.

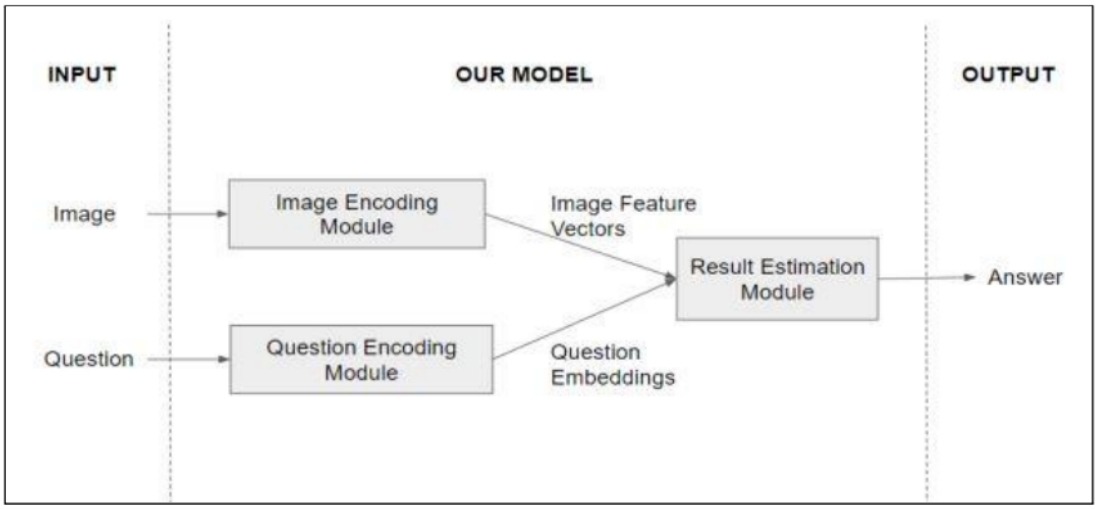


Figure 6.4

6.2 Low Level Design:

The Section deals with the lowest level dissection of the high level design developed in Phase 1. Here we delve into the modular and unit components that constitute the development of this tool.

6.2.1 Overview

The below diagram Summarises the high level design that we intended to do.

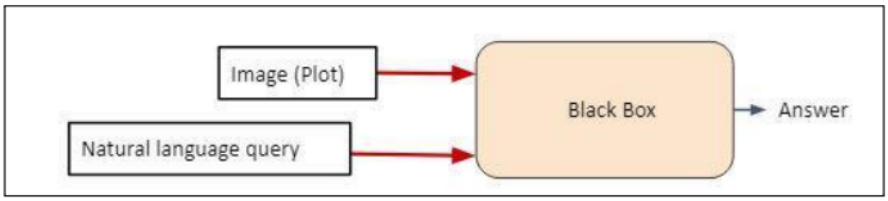


Fig 6.5 : Proposed High level View of The VQA system

In the figure we have two inputs being fed into the Black Box which are an input image [the graphical plot in scope] and a Natural language query related to the graphical image. whereas in the lower level design the black box is further expanded and cut open. The Black Box Constitutes the core of the visual question answering system.

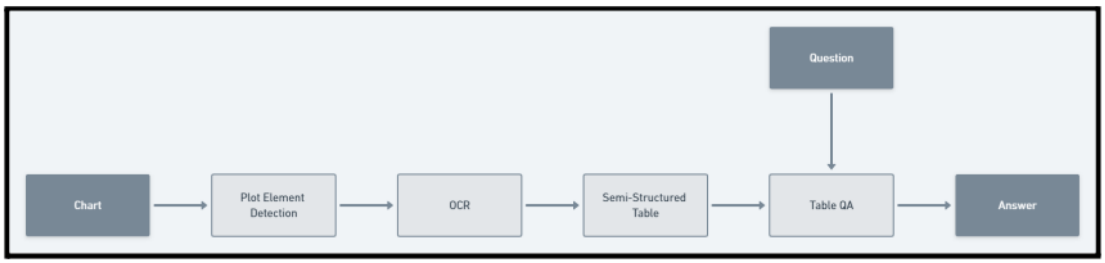


Fig 6.6 The cut open view of the black box and deep delve into the modules

There are 4 stages within the Black Box as shown in ***Fig 6.6***

- **The plot element detection stage**
- **The optical character recognition stage**
- **The Semi-structured table generation stage**
- **Table Question Answering Stage**

The inputs to black-box/ VQA system

- A graphical plot [bar {vertical/horizontal/grouped} , dot , line]
- A question posed [in vocab / out of vocab] related to the plot image

The phase wise dissection will be done in the following sections.

6.2.2 Purpose

The purpose of the low level design document is to provide a detailed description about the working of each and every unit and how they all work in union to achieve the desired result. It is used by designers, operation teams, implementers/dev and dev-test members . This will serve as a documentation for developing stubs/drivers .

Here we provide a description about the four stages within the Black Box and how they inter communicate with each other and interface with each other. Moreover this is the phase of a project in

which the application logic is designed and ready to be implemented. The exit criteria / input criteria - data to every phase needs to be dissected and presented in detail.

6.2.3 Scope

The scope of this subchapter is to address the flow of information through the pipeline and stage wise requirement which is needed to accomplish the goals of corresponding stages. This implementation of VQA on statistical plots takes into consideration **plots of type = {Dot , Line , Bar[Hbar , Vbar , Grouped]}** and **Questions = {Open-ended, In-vocabulary} .**

Overview of the tools that have significant contribution in every stage is provided. Alongside constraints , dependencies and assumptions existing between the modules/stages is also discussed. An overview regarding the novel practices and new ideas infused within the system is also discussed along with examples and variants of those.

6.2.4 Design Constraints, Assumptions, and Dependencies

The Environment , hardware software dependencies needed to run this pipeline

The training environment specification is as follows

- **Platform** : Google Colab Pro (Cloud)
- **RAM** : 26GB
- **Disk** : 110GB (Cloud)
- **GPU** : 16GB , Tesla - T4
- **Training Data Size** : 6.x GB
- **Test Data Size** : 1.5xGB

Assumptions of the model/VQA system

- ➢ The VQA is limited to only certain class of graphs and questions
- ➢ The input image to the model should be one among **{Dot , Line , Bar[Hbar , Vbar , Grouped]}**
- ➢ The questions posed should be of type **{Open-ended, In-vocabulary}.**

- The type of questions can be based on arithmetic mean , median , difference , sum , data retrieval . comparison or boolean.

Structural questions are not handled within the scope of this implementation.

Dependencies between the stages and the Input/Exit data+Criteria

Stage	Input Criteria	Exit Criteria	Output
Plot element detection	Input Plot Image belonging to certain class of graphs	Bounding box annotation around the plot elements.	A text tabular formatted equivalent of a json file , holding the coordinates of the plot elements[topleft_x , topleft_y,bottom_x , bottom_y] and the confidence that the element belongs to a class
OCR	text tabular file from previous stage + Image	use OCR module like tesseract to read the character within the bounding coordinates in the image	extracted texts from the bounding box specific region according to the category of the plot element
Semi-structured table generation	textual data extracted from OCR	format the data into a semi-structured table on which queries can	A CSV file which is a tabular format

		be executed	of the graph input image
Table Question Answering	The CSV file/tabular format of the graph + Question	classify the queries into boolean vs data-retrieval Execute the queries on the table and accumulate answer	The final answer to the input question

Table 6.1

We can clearly see the **linear dependency that exists across the modules** , failure of any one of the intermediate modules will tend to have a cascading effect on the follow up stages.

There also exists few dependencies on the modules/of the shelf components that are being made use of in every stage.

Dependencies on the Modules/libraries and packages used

- Detectron-2 [5]
- Pytorch - 1.9.0
- Tesseract [6] , conda environments
- cv2 , TAPAS [7] , TABFACT & SEMPRE
- All other utility modules like OS , JSON , NUMPY , CSV , Scipy etc.

Constraints regarding the questions posed

Data retrieval , sum , average , columnar max- min questions handling capability was already built into TAPAS whereas we have added custom methods/operators handling methods to accept a variety of questions based on range , quartile , difference etc , these are based on break up over a particular keyword.

6.2.5 Module Wise Design Description

A) Plot Element Detection Stage

- Input : Image{graphical plot}
- Output : formatted text file derived from JSON
- The plot elements of an input image are extracted by training an object detection model over a large collection of samples {images + annotations} .
- **Detectron-2** is faster , flexible and vast in terms of configuration,models and implementation due to its API availability when compared with its parent , hence we use it as the object detection and bounding box generation tool.
- Few samples are shown below

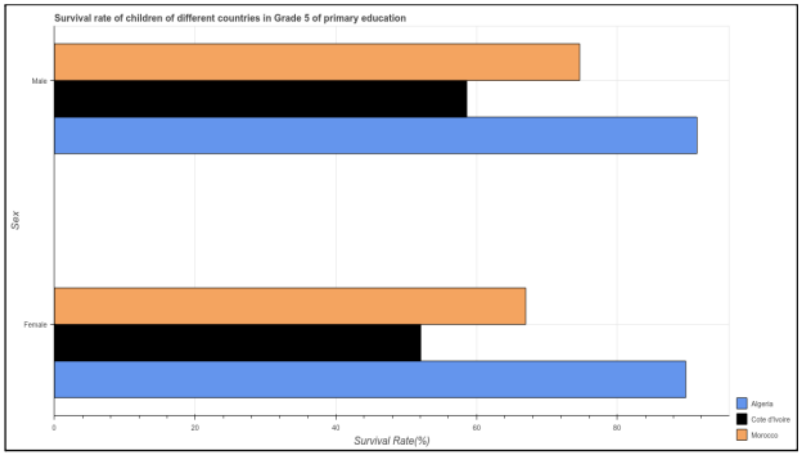


Figure 6.7 A horizontally grouped bar graph {Test-input}

The image along with its annotation is passed onto a model trained by us , we now have the weight of the model saved for further testing and inference , as the image passes through the designed object detection model , we expect bounding boxes to be drawn around every potential plot element which was learnt by the model. The output of a model will be a json file which maps all the detected objects (plot elements) to the **class** to which it belongs , with an appropriate **confidence value** and its **bounding box tensor**.

```
{'instances': Instances(num_instances=22, image_height=650, image_width=1245, fields=[pred_boxes: Boxes(tensor([[  27.2519,  459.9764,   63.6584,  474.0742],
        [  40.1922,  102.9239,   64.2441,  117.1245],
        [1173.8287,  619.1683, 1215.5663,  639.1750],
        [  74.6989,  109.7839,  726.3925,  163.6040],
        [ 735.3030,  612.0875,  747.5630,  625.9379],
        [  79.2105,  521.1633, 1080.1464,  574.4030],
        [  75.5007,  467.3218,  653.8602,  521.5388],
        [  79.1654,   56.3122,  898.4966,  110.0806],
        [  76.8082,  414.3173,  815.8287,  467.9611],
        [1174.1018,  596.1894, 1235.4202,  616.2552],
        [1173.9548,  572.9149, 1208.1724,  592.9194],
        [1148.6736,  572.9440, 1168.8115,  593.0646],
        [1148.6539,  619.1230, 1168.7286,  638.9912],
        [  76.3105,  164.4282, 1101.3408,  217.3091],
        [ 512.8422,  611.9535,  524.9588,  625.9399],
        [ 290.6320,  612.0994,  303.0508,  626.0181],
        [ 546.7985,  629.4926,  666.7787,  647.7834],
        [  72.1504,  612.0005,   78.2134,  625.9789],
        [ 956.0478,  612.1361,  968.4012,  626.0216],
        [1149.1019,  595.9470, 1169.0248,  616.4178],
        [  76.6021,    9.0043,  769.3833,   27.0149],
        [   6.1733,  301.8087,   24.0409,  330.2988]], device='cuda:0')), scores: tensor([1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 0.9999,
        0.9999, 0.9999, 0.9999, 0.9999, 0.9999, 0.9999, 0.9999, 0.9999, 0.9999,
        0.9999, 0.9999, 0.9997, 0.9996], device='cuda:0'), pred_classes: tensor([9, 9, 2, 0, 7, 0, 0, 0, 0, 2, 2, 4, 4, 0, 7, 7, 6, 7, 7, 4, 5, 8],
       device='cuda:0'))])
```


Figure 6.7 The json output produced by detectron tester , which maps object elements to its class , with a certain confidence and the bounding box tensor

Given the classes to which the plot elements belong span from [0-9] , we need a mapping between the class numbers and the plot elements , which is provided in the ***Figure 6.8*** below.

```
mapping = {  "bar": 0,
             "dot_line": 1,
             "legend_label": 2,
             "line": 3,
             "preview": 4,
             "title": 5,
             "xlabel": 6,
             "xticklabel": 7,
             "ylabel": 9,
             "yticklabel":9
           }
```

Figure 6.8

Now that we have the mapping and the unstructured json data , it is better to have them both combined and produce a readable and easily understandable textual format that can be used for further processing in the pipeline.

```
class confidence       top_x            top_y             bottom_x          bottom_y
bar 0.9999309778213501 76.31050872802734 164.42819213867188 1101.3407821655273 217.30908203125
bar 0.9999496936798096 76.8082275390625 414.31732177734375 815.82861328125 467.9610595703125
bar 0.9999642372131348 74.69889831542969 109.78387451171875 726.3924407958984 163.60403442382812
bar 0.9999502897262573 79.16533660888672 56.312198638916016 898.4965744018555 110.08055877685547
bar 0.9999561309814453 79.21051025390625 521.163330078125 1080.1463623046875 574.4030151367188
bar 0.9999512434005737 75.50065612792969 467.32183837890625 653.8601531982422 521.538818359375
legend_label 0.9999395608901978 1173.954833984375 572.9148559570312 1208.17236328125 592.91943359375
legend_label 0.9999446868896484 1174.101806640625 596.1893920898438 1235.420166015625 616.2551879882812
legend_label 0.9999701976776123 1173.8287353515625 619.1683349609375 1215.5662841796875 639.175048828125
preview 0.9999374151229858 1148.673583984375 572.9439697265625 1168.8115234375 593.0646362304688
preview 0.999933123588562 1148.6539306640625 619.1229858398438 1168.7286376953125 638.9912109375
preview 0.9998660087585449 1149.1019287109375 595.9469604492188 1169.0247802734375 616.4177856445312
title 0.9996670484542847 72.77204246520996 8.644117126464844 775.538344116211 28.365630197525025
xlabel 0.9998942613601685 519.4585968017578 629.4925537109375 672.112916015625 647.783447265625
xticklabel 0.9998867511749268 72.15039825439453 612.00048828125 78.21342468261719 625.9788818359375
xticklabel 0.9999247789382935 512.8422241210938 611.9534912109375 524.9588012695312 625.9398803710938
xticklabel 0.9999061822891235 290.6319580078125 612.0994262695312 303.05084228515625 626.0181274414062
xticklabel 0.9999575614929199 735.3030395507812 612.0874633789062 747.5630493164062 625.9378662109375
xticklabel 0.9998866319656372 956.0477905273438 612.1361083984375 968.4012451171875 626.0216064453125
ylabel 0.9995629191398621 5.8646334409713745 289.736396484375 24.233238487243653 346.813737487793
yticklabel 0.9999923706054688 40.192203521728516 102.92390441894531 64.24410247802734 117.1244888305664
yticklabel 0.9999945163726807 27.25185203552246 459.97637939453125 63.65839195251465 474.07421875
```

Figure 6.9 : properly formatted text file which has the bounding box tensors of all the plot elements that were detected in the model

As seen above , all the bounding boxes corresponding to the plot elements have been extracted according to their classes .We chose **Faster-RCNN** as our object detection model which is a **descendant from the R-CNNs series**.The heuristic behind doing so is due to the presence of the RPN (Regional proposal network is known for locating feature targets accurately) and the inference time.
We also need to decide on the **feature extractor model** for serving as the **backbone/feature-extractor** for our faster-RCNN Setup.We Chose **Resnet-101** to be our feature extractor due to **skip connections** and **residual blocks** which can reduce the problem of vanishing gradients in a very deep network like Resnet-101.

B) OCR detection Stage

- The textual format of the json file as shown in ***Figure 6.9*** is passed into this stage .
- The images directory is made accessible to this module.
- With the help of bounding box coordinates extracted , we can accurately and locally capture the text information within the bounding boxes rather than passing an entire image into the OCR module .
- The captured text is then read using OCR and classified into its category.

C) Semi Structured table generation

This phase is the crucial phase of converting the graphical data to its tabular format based on the OCR readings done in accordance with classes. We'll walk through an example to explain the conversion of an image (plot) into a tabular data format (csv). An input graph goes through several transformations in the pipeline but not necessarily in the image format; right after the OCR stage , the input image is no longer required , the image is discarded and its tabular csv format comes into play for the final table QA stage.

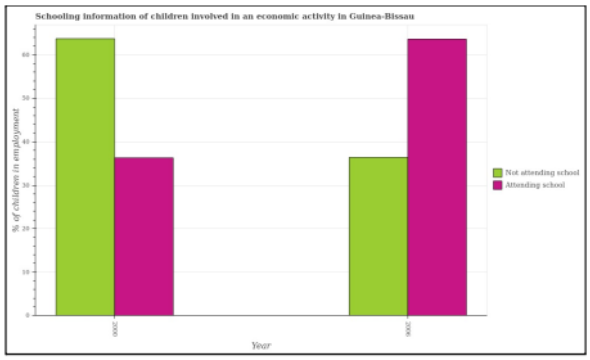


Figure 6.10 : A Simple Bar Graph

The simple bar graph goes through the plot element detection stage , and a text file emerges as the outcome as discussed in the previous stage.This text file will then be passed onto the OCR stage along

with the image to extract the textual content within the boxed coordinates.Following which the information is structured in the semi structured table generation phase.

```
Category   Score            left_x           top_y            right_x          bottom_y
yticklabel 0.9999904632568359 27.946645736694336 246.9253387451172 41.919342041015625 260.8976135253906
yticklabel 0.9999895095825195 27.873197555541992 328.9063720703125 41.86872100830078 343.05303955078125
bar 0.9999873638153076 640.291259765625 284.0574645996094 749.989990234375 580.9351806640625
bar 0.9999823570251465 201.92819213867188 281.8130798339844 312.2918701171875 578.4205322265625
legend_label 0.9999779462814331 932.5986328125 303.1197509765625 1071.5126953125 322.6226806640625
yticklabel 0.9999773502349854 27.868160247802734 409.8987121582031 41.989967346191406 423.9773254394531
bar 0.9999696016311646 750.8607177734375 62.46775817871094 861.982421875 581.0493621826172
xticklabel 0.9999669790267944 194.113037109375 590.9034423828125 208.31503295898438 618.8867797851562
yticklabel 0.9999643564224243 34.982704162597656 572.9815063476562 41.99324035644531 587.0731201171875
bar 0.9999592304229736 91.97103881835938 59.920265197753906 202.525070190429 7 581.5347671508789
preview 0.9999498128890991 908.1177368164062 303.0318908691406 928.0110473632812 322.7835998535156
yticklabel 0.9999486207962036 28.012062072753906 164.76251220703125 42.00925827026367 178.90281677246094
yticklabel 0.9999397993087769 28.047203063964844 492.2265625 42.02458572387695 505.870849609375
xlabel 0.9999388456344604 457.864501953125 627.0880737304688 493.36260986328125 646.0440673828125
xticklabel 0.999916672706604 743.106201171875 590.95849609375 756.9241943359375 619.1182861328125
legend_label 0.9998894929885864 932.1422119140625 325.8930969238281 1043.439697265625 345.8262023925781
preview 0.9998648166656494 908.1642456054688 326.23406982421875 928.19921875 346.0467834472656
yticklabel 0.9998058676719666 28.096742630004883 84.25582885742188 42.00232696533203 97.83050537109375
ylabel 0.9998049139976501 6.735439777374268 190.0906524658203 26.02379274368286 425.5273742675781
title 0.9994369149208069 49.137367248535156 9.029414176940918 796.2882461547852 27.020319938659668
```

Figure 6.11 : Textual output of input graph

	Year	Attending school	Not attending school
0	2000	37.06615560158244	64.78018619662012
1	2006	64.48456969920525	37.0474010349917

Figure 6.12 : Tabular CSV format of the input graph

Now this csv file would be further used in the table question answering phase.

D) Table Question Answering Stage

- We have made use of **Google's TAPAS** to answer questions from tabular data
- Tapas selects a subset of table cells and applies aggregation/retrieval operations on top of them
- It extends BERT architecture with additional embeddings that capture tabular structure, and with two classification layers for selecting cells and predicting a corresponding aggregation operator.
- We have added our custom operations and methods to suit the desired output.
- It is trained on Wikipedia Tables and provides a pre-trained model for end tasks.

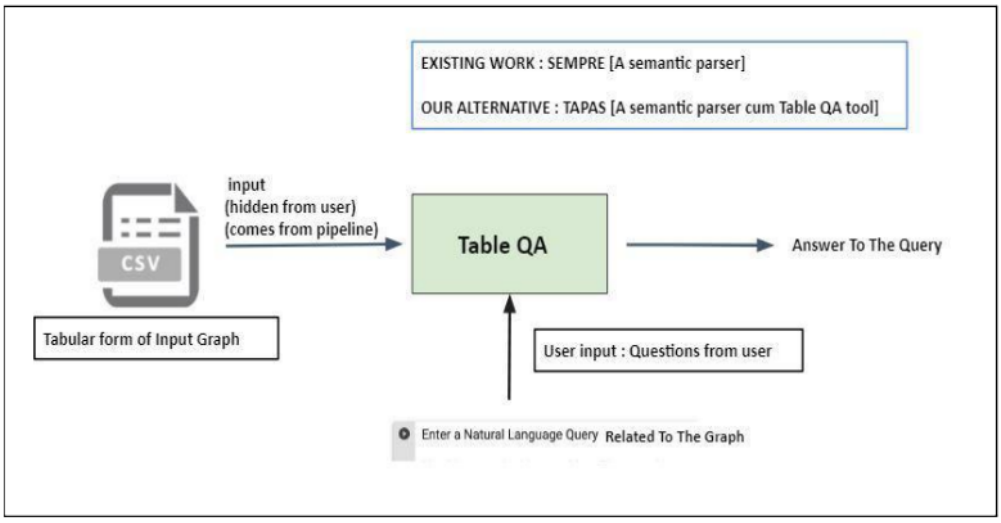


Figure 6.13 : The Table Question Answering Stage

CHAPTER 7

Proposed Methodology

The following sections comprise the well detailed idea about the working of the pipeline designed for this work . It consists of the novelty induced by us in this work . the inspiration for this work and the key aspects of the model.

Stage 1 : The Plot Elements Detection

- From the previous sections it is certain that the primary step is to detect the possible plot elements , every single element needs to be captured , hence we adopt an object detection tool called the detectron , we choose the successor of the detectron , which is detectron-2
- Detectron-2 has a vast collection of models in its model zoo which provides us a flexible option to choose the baseline and backbone networks at ease by keeping the speed/accuracy trade-offs in consideration.
- The input dataset is a collection of training images and their corresponding annotation file , they reside separately . They must be correlated together before they are ready for training. This is easily achieved by detectron-2's dataset catalog and metadata set catalogue.
- The choice of the model has been discussed in the previous system design section . The output of this stage is a json file which is converted into a structured text file.

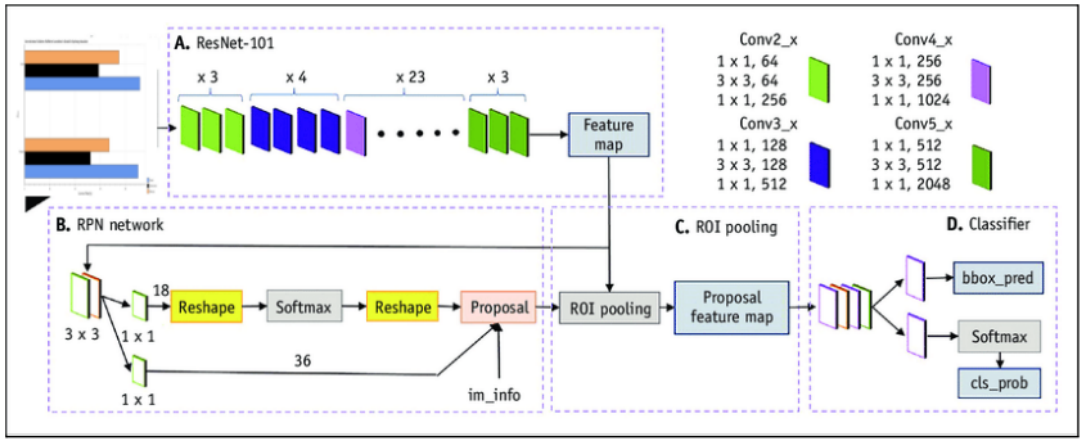


Figure 7.1 The illustration of an image being fed into a faster-RCNN with Resnet 101 backbone

Given the classes to which the plot elements belong span from [0-9] , we need a mapping between the class numbers and the plot elements , which is provided in ***Figure 6.8*** .

Stage 2 : The Optical Character Reader/Recognition Stage

- There are a total of 10 different plot elements that can be found in any statistical plot. These can be grouped into two categories : Textual Elements and Visual Elements
- Textual elements correspond to the title of the plot, y-axis label, x-axis label, x-tick, y-tick values and the labels corresponding to the legend
- To read the textual and numeric data off the textual components, we make use of the formatted textual output from the previous stage, and an Optical Character Recognition module
- The OCR module used is pyocr which is a wrapper for the Tesseract OCR engine
- Detected textual elements are cropped to the bounding box size (which is obtained from the previous stage), then converted to gray-scale and passed onto the pyocr module.
- Thus, the output of this stage is textual data corresponding to the detected textual elements.

Stage 3 : Semi Structured table generation

- The output of this stage is a semi-structured table that encapsulates all of the data in the statistical plot.
- For the textual elements, we have already obtained textual data. This stage is responsible for mapping legend values to the legend color, x-ticks to the x-axis label and the y-ticks to the y-axis label. This is done by associating the legend/x-tick/y-tick value bounding box to the closest legend color/ x-axis/ y-axis boundary respectively.
- For the visual elements, each element is associated with an axis, and a corresponding legend. The color of the visual element is matched with the legend colors, and the legend of the closest match is associated with the element. To find the value associated with the bar, the information of height is taken from the bounding box representation, and the closest y-tick is mapped.
- Doing this for all visual elements will fill the table.

Stage 4: Table Question Answering Stage

- Given a semi-structured table and a relevant natural language question as input, this stage is responsible for producing an answer to the question from the table as an output.
- The questions can be classified into two types. The first type corresponds to open-ended questions that have an unrestricted answer domain. The second type corresponds to questions that require a Yes/No (binary) answer.
- To handle open-ended questions, we have made use of the existing TaPas (Table Parsing) model. This model is based on the BERT's encoder with certain modifications. Positional embeddings are used to encode tabular data, and two additional classification layers are introduced to select cells of the table and the aggregation operation to be performed.
- Our work makes use of a pre-trained TaPas model that has been trained on the WikiTables Questions dataset with intermediate pre-training. This model can handle 3 types of aggregation operations - SUM, COUNT, AVERAGE. To add to the capabilities of this model, we have added other operations such as RATIO, DIFFERENCE, MEDIAN, TREND, RANGE and QUARTILES.
- To handle questions that require a Yes/No answer, we have used a TaPas model trained on the TabFact dataset. This is a dataset used for table entailment and fact verification. We have extended its capabilities by adding other operations like in the earlier mentioned model
- An important aspect here is that given an input question we would need to know what type of question it is (whether it is an open-domain question or a yes/no question). For this, we have implemented a binary question classifier.

Binary Classifier

- There are two categories of questions addressed: Open-ended and Yes/No. Each of this is an independent model and hence to integrate into a single pipeline, a binary classifier is used.
- Binary classifier model classifies the given input question into Yes/No class (class 0) or Open-ended class (class 1)
- A dataset is prepared for this purpose from the PlotQA data. The model is trained on questions of all types of plots (i,e, vbar_categorical, hbar_categorical, dot_line, line) and the answers which are converted into the categories of 0 or 1

- The Deep Learning model is trained to classify the input questions into correct classes. The following are the model specifications:
  - Embedding Type: GloVe
  - Max Pooling Layer: 1
  - Dense Layer: 10 nodes and ReLu activation
  - Dense Layer: 1 node and Sigmoid activation
  - Optimizer: Adam
  - Loss Function: Binary Cross-Entropy
  - Metric: Accuracy

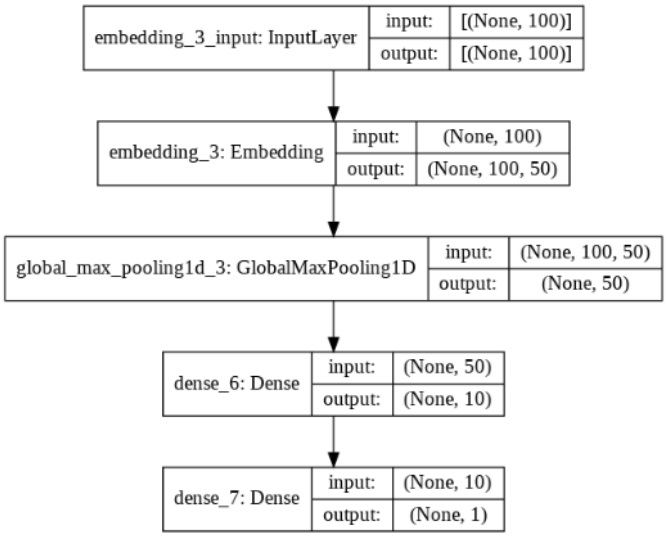


Figure 7.2

- The model was trained for 10 Epochs in a batch size of 10. Accuracy of 1.0 was obtained.

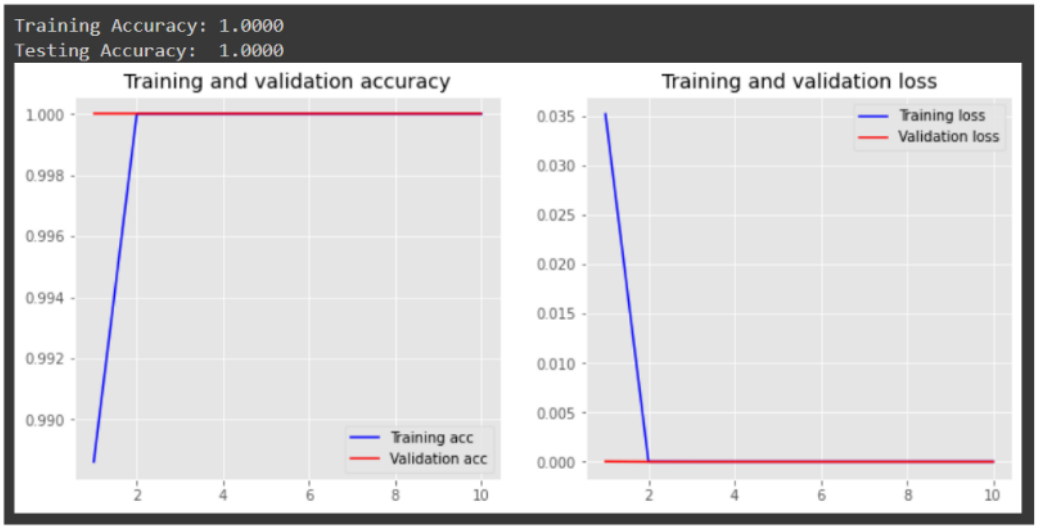


Figure 7.3

- The trained model was saved and loaded to classify test images. If the model outputs class 0, the question is passed to TABFACT model and if the model outputs class 1, the question is passed to TAPAS model

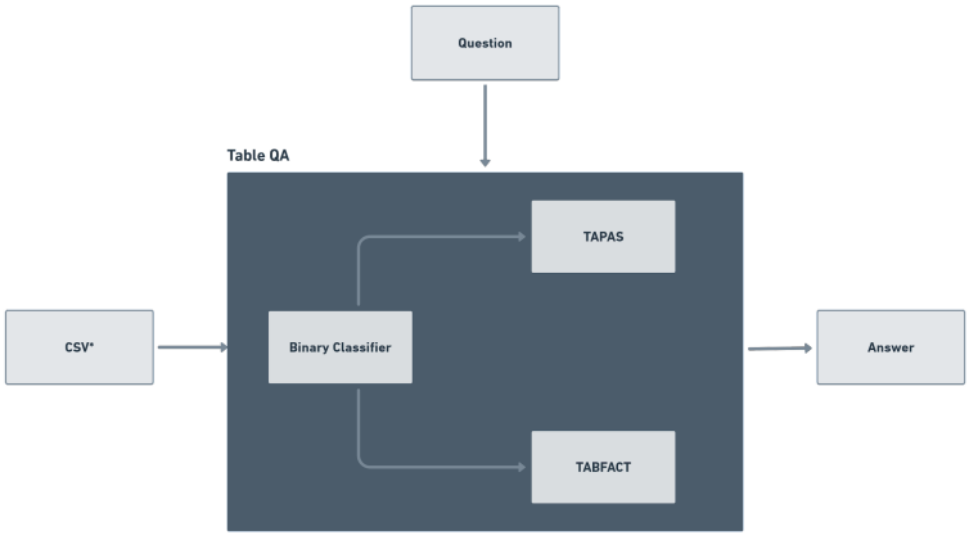


Figure 7.4 Table Question Answering Model

CHAPTER 8

Implementation and Pseudocode

This section details the implementation details and a pseudocode for each of the modules in our pipeline.

Stage 1 : The Plot Elements Detection

To detect the elements in the plot and map it to its bounding box representation, we have made use of Detectron 2 - an object detection tool with a bounding box generator.

The training data includes the training images and the annotations from the PlotQA dataset. We correlate the image with its corresponding annotation and pass it to the Detectron trainer. We chose the object detection model to be the Faster-rcnn and Resnet_101 as the backbone network. The model was trained for 2 lakh iterations on a learning rate of 0.0004, with a decay of 0.1.

On all of the test images, the model was tested to generate the bbox representations as a JSON file, which was then formatted to a text file.

```
#TRAINING

Load (train_images , Annotation_file)
model = Trainer()
dataset : Split(train_images , Annotation_files , Splits = 3)
model.data : Correlate_dataset_with_annotations(dataset)
sample(data) #check If Image And Annotation Are Corresponding
choose Object_detection_model And Backbone_network
model.object_detection_model : Faster-rcnn
model.backbone_model : Resnet_101
iters : 21 ; Lr : 0.0025 ; Gamma : 0.1
model.train(data , Iters , Lr , Gamma Object_detection_model , Backbone)

#TESTING

Load(test_images)
model.test_data :Register_for_testing(test_images)
json_result : model.test(generate_json_result = True)
txt_format : convert_json_o_text(json_result)
```

Figure 8.1 Pseudocode of the Plot Elements Detection Stage (Detectron - 2)

```
from detectron2.data.datasets import register_coco_instances
#split-1
register_coco_instances("Train_A", {}, "/root/Work/Data/PlotQA/annotations/train_50k_annotations.json", "/root/Work/Data/TRAIN/png")
#implies that images from the specified path and the annotation from the specified path must be correlated and superimposed before training

#split-2
register_coco_instances("Train_B", {}, "/root/Work/Data/PlotQA/annotations/train_50k_1l_annotations.json", "/root/Work/Data/TRAIN/png")

#split-3
register_coco_instances("Train_C", {}, "/root/Work/Data/PlotQA/annotations/train_1l_end_annotations.json", "/root/Work/Data/TRAIN/png")
```

Figure 8.2 Registering The Data Splits For training (Image+Annotation{json})

```
config.merge_from_file(model_zoo.get_config_file("COCO-Detection/faster_rcnn_R_101_FPN_3x.yaml"))

config.DATASETS.TRAIN = ('Train_A' ,'Train_B' , 'Train_C')

config.DATASETS.TEST = ()

config.NUM_GPUS = 1

config.DATALOADER.NUM_WORKERS = 4

config.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-InstanceSegmentation/mask_rcnn_R_101_FPN_3x.yaml")  # Let training initialize from model zoo

config.SOLVER.BASE_LR = 0.0004                                          #Kept minimal , so that global optimum can be hit during gradient descent

config.SOLVER.MAX_ITER = 200000                                          #Must Change for future training , similar to epoch

config.SOLVER.STEPS = [1100,40000,120000]                                        #Stages at which LR Reduction must occur

config.SOLVER.GAMMA = 0.1                                               #Reduction factor for LR

config.SOLVER.IMS_PER_BATCH = 1                                         #Images seen by GPU per sec

config.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 512                       #Batch-size

config.MODEL.ROI_HEADS.NUM_CLASSES = 11                                 #10 instances + 1 background

config.OUTPUT_DIR = './Output_2L_Cont'

#config.TEST.EVAL_PERIOD = 30000

config.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.7
```

Figure 8.3 The training yaml configuration file customized as per our LLD

```
from detectron2.engine import DefaultTrainer

trainer = DefaultTrainer(config)
trainer.resume_or_load(resume=False)
trainer.train()
```

Figure 8.4 The training Stage

Stage 2 : The Optical Character Reader/Recognition Stage

We used the pyocr tool (a wrapper for the Tesseract Engine) for optical character recognition. This was used for all the textual elements detected in the previous stage.

```
22  ------------------------------------------------------------------------------
23
24
25  ocr : OCR()
26  ocr.load_utilities()
27  ocr_extractions : ocr.load(txt_format  , input_image)
28
29
30  ------------------------------------------------------------------------------
```

Figure 8.5 Pseudocode of the OCR Stage

Stage 3 : Semi Structured table generation

The output of this stage is a semi-structured table in the form of a CSV. The textual and numeric information extracted from the ocr is populated into a table. The values (height information in the case of a bar plot, x and y coordinates in case of a line and dot-line plot) are filled using the logic mentioned in the proposed methodology section.

```
CSV_maker : CSV()
CSV_maker.ocr_readings : ocr_extractions()
Tabular_format : CSV_maker.generate_table_csv()
```

Figure 8.6 Pseudocode of Semi-structured Table Generation

Stage 4: Table Question Answering Stage

We have made use of the existing TaPas (Table Parsing) module for the task of question answering from tables. The csv obtained from the previous stage is pre-processed. This is the stage where the

questions entered by the user are processed. As mentioned earlier, the questions are classified into either an open-ended question or a question that requires a Yes/No answer. If the question is an open-ended question, it is passed onto the TaPas model pre-trained on the WTQ dataset, otherwise it is passed on the TaPas model trained on the TabFact table entailment dataset. The output here is an answer to the question.

```
table_qa : TAPAS()
csv_table : preprocess(Tabular_format)
table_qa.table : csv_table
table_qa.queries : List(Read_from_users())
table_qa.queries.classify()
table_qa.answers()
```

Figure 8.7 Pseudocode of Table Question Answering

AN END-TO-END EXAMPLE

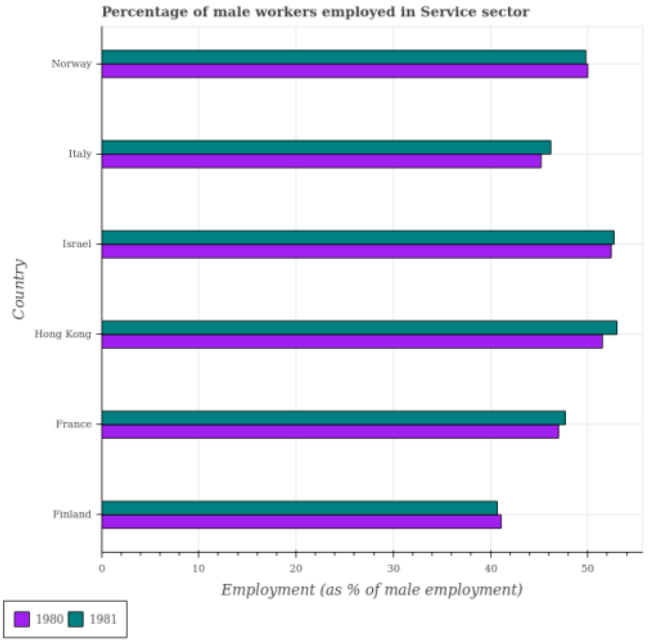


Figure 8.8 Input Image

Input : An image and a question

Output : An answer to the question

Given the input image as shown in ***Figure 8.8***, we demonstrate each of the stages below.

Stage 1 : The Plot Elements Detection

The output of this stage is a formatted text file that contains the bounding box representations of each of the elements of the plot represented in ***Figure 8.9***. This statistical plot contains a total of 31 plot elements. Therefore, the output shows a txt file with 31 lines, each corresponding to a plot element.

5574.txt

```
yticklabel 0.9999946355819702 70.65868377685547 166.0301055908203 95.9031753540039 180.24429321289062
yticklabel 0.9999940395355225 56.70259094238281 460.2558288574219 96.12216186523438 473.992431640625
yticklabel 0.9999926090240479 52.40139389038086 68.11477661132812 96.11300659179688 81.90919494628906
yticklabel 0.9999892711639404 64.40562438964844 264.47857666015625 96.38269805908203 278.0016174316406
bar 0.9999853372573853 107.04766845703125 565.3331298828125 539.787109375 580.1272583007812
bar 0.9999684095382690 106.83328247070312 74.97222900390625 635.513916015625 89.83660125732422
preview 0.9999650716781616 10.059914588928223 668.806884765625 29.93454360961914 688.8609619140625
bar 0.9999635219573975 108.80455017089844 354.7571105957031 669.4835205078125 369.15228271484375
yticklabel 0.9999619722366333 53.4467887878418 558.0203857421875 96.28006744384766 572.0387573242188
bar 0.9999618530273438 105.96603393554688 369.0704650878906 651.154052734375 384.01531982421875
preview 0.9999606609344482 69.05435180664062 668.9619750976562 88.89054870605469 689.03515625
bar 0.9999605417251587 106.07267761230469 270.9334411621094 661.7080688476562 285.9010314941406
bar 0.9999490976333618 107.0818099975586 467.1742248535156 604.5418701171875 482.14801025390625
bar 0.9999490976333618 107.93446350097656 256.6279296875 669.5387573242188 271.04522705078125
bar 0.9999490976333618 107.84508514404297 172.93421936035156 584.4448852539062 187.95010375976562
xticklabel 0.9999262094497681 523.0034179687500 617.0694580078125 536.9373168945312 631.1053466796875
bar 0.9999213218688965 107.91783905029297 550.761962890625 536.0125732421875 565.3912963867188
legend_label 0.9999172687530518 34.85472106933594 668.88671875 65.8724365234375 689.089599609375
legend_label 0.9999128580093384 93.74087524414062 669.108642578125 124.3467025756836 689.2234497070312
xticklabel 0.9999068975448608 104.04006958007812 616.9739990234375 110.94221496582031 630.9584350585938
xticklabel 0.9999068975448608 204.91419982910156 617.0042114257812 218.94931030273438 631.0257568359375
xticklabel 0.9998940229415894 417.209716796875 617.04736328125 430.898681640625 631.0352172851562
xticklabel 0.9998908042907715 310.7530517578125 617.034423828125 325.072265625 630.994384765625
bar 0.9998842477798462 109.16693115234375 452.7776794433594 612.639892578125 467.078369140625
xticklabel 0.9998824596405029 627.995849609375 616.922119140625 642.1049194335938 630.9342041015625
bar 0.9998799562454224 107.85015869140625 60.325401306152344 634.2732543945312 75.0029296875
bar 0.9998627901077270 109.2906494140625 158.4036865234375 593.9943847656250 172.654296875
yticklabel 0.9998502731323242 33.24961853027344 361.7756042480469 96.37665557861328 376.0375061035156
ylabel 0.9997543692588806 7.122566223144531 286.8748779296875 26.106121063232422 352.16131591796875
xlabel 0.9997450709342957 238.1763916015625 637.1104736328125 565.7171020507812 656.1470336914062
title 0.9996367692947388 106.48019409179688 8.978804588317871 514.6640014648438 27.022794723510742
```

Figure 8.9 Output of Detectron - 2

Stage 2 & 3: The Optical Character Recognition and Semi Structured table generation stage

The bounding box representations are mapped onto the image to obtain textual and numeric information, which is then populated into a tabular format.***Figure 8.10*** shows the CSV output obtained for the image in ***Figure 8.8***.

	Country	1980	1981
0	Finland	41.56915064054187	41.21788389031577
1	France	48.97010591234	48.40140328049439
2	Hong Kong	53.624025195560996	54.12660380966694
3	Israel	54.191569798758344	54.5127801772846
4	Italy	46.373520540205824	46.33205263996708
5	Norway	51.48881822230819	51.018932095367745

Figure 8.10 Output of Stage-3 (CSV)

Stage 4: Table Question Answering Stage

This stage makes use of the CSV as seen in ***Figure 8.10*** to answer the question posed by the user. The types of questions that can be addressed by our model are as follows.

1. **Count**

   Q: What are the total number of countries ?
   A: 6

2. **Sum**

   Q: What is the total number of male workers employed in 1980 ?
   A: 296.2171903097152

3. **Average**

   Q: What is the average number of male workers in the year 1980 ?

A: 49.36953171828586

Q: What is the average number of male workers in the year 1981 ?
A: 49.26827598218275

4. **Minimum**

Q: across all countries, what is the minimum percentage of male workers employed in service sector in 1980 ?
A: 41.56915064054187

5. **Maximum**

Q: across all countries, what is the maximum percentage of male workers employed in service sector in 1980 ?
A: 54.191569798758344

6. **Difference**

Q: What is the difference between the average number of male workers employed for the year 1980 and 1981 ?
A: DIFFERENCE = 0.10125573610311278

Q: What is the difference between the number of male workers employed for the country france in 1980 and 1981 ?
A: DIFFERENCE = 0.5687026318456105

- Keyword = "difference between"
- The two entities must be separated by "and"

7. **Median**

Q: What is the median number of male workers employed in the year 1980 ?
A: MEDIAN = 50.22946206732409

- Keyword = "median"
- Column name for which the median has to be found

8. **Ratio**

Q: What is the ratio of male workers employed in 1981 to 1980 for the country hong kong ?
A: RATIO = 1.0093722657385955

- Keyword = “Ratio“
- QUANTITY_1 "to" QUANTITY_2

9. **Trend (Increasing or Decreasing)**

Q: What is the trend of male workers employed for the countries finland, france, hong kong in 1980 ?
A: TREND = INCREASING

Q: What is the trend of male workers employed for the countries israel, italy in 1980 ?
A: TREND = DECREASING

Q: What is the trend of male workers employed for the countries israel, italy, norway in 1980 ?
A: TREND = NONE

- Keyword = “trend“
- List of comma separated entries followed by "in" COL_NAME

10. **Selection operation on cell**

Q: What is the number of male workers employed for country france in the year 1981 ?
A: 48.40140328049439

11. **Select operation on cell after applying aggregation operation**

Q: Which country has the minimum number of male workers employed in the year 1981 ?
A: Finland

12. **Selection and Aggregation operation on subset of rows**

Q: What is the sum of male workers employed for the countries france, finland in the year 1980 ?
A: 90.53925655288188

Q: What is the average number of male workers employed for the countries france, finland in the year 1980?
A: 45.26962827644094

Q: What is the maximum number of male workers employed for the countries france, finland in the year 1980 ?
A: 48.97010591234

13. Project operation on column

Q: What are the names of all the countries ?
A: Finland, France, Hong Kong, Israel, Italy, Norway

Q: list out the countries
A: Finland, France, Hong Kong, Israel, Italy, Norway

14. Range

Q: What is the range of % of male employment for the year 1980 ?
A: RANGE = 12.622419158216474

15. Quartiles (Q1 and Q3)

Q: find the quartiles for the year 1980
A: FIRST QUARTILE (Q1) = 43.97133559037385
SECOND QUARTILE (Q2) = 50.22946206732409
THIRD QUARTILE (Q3) = 53.90779749715967

16. IQR

Q: find the interquartile range for the year 1980
A: INTER-QUARTILE RANGE = 9.936461906785823

17. Structural Query

Q: What is the title of the graph ?
A: TITLE OF THE GRAPH = Percentage of male workers employed in Service sector

Q: What is the label or title of the x-axis ?
A: X-LABEL = Employment (0S % of male employment)

Q: What is the label or title of the y-axis ?
A: Y-LABEL = Country

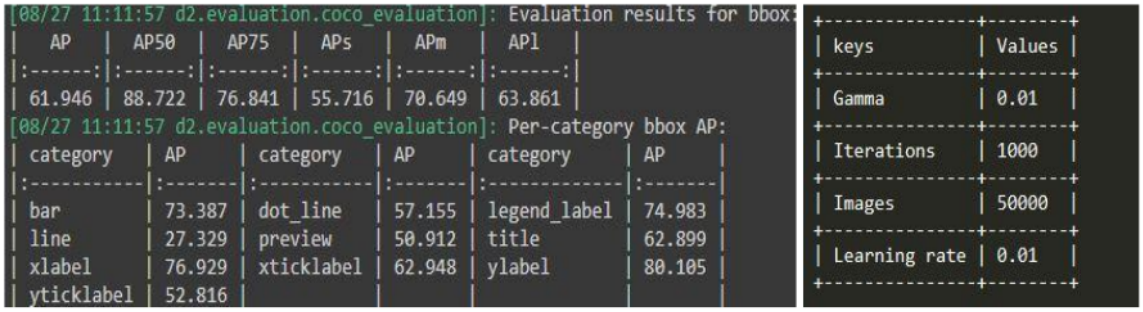
CHAPTER 9

RESULTS AND DISCUSSION

The steps mentioned in our methodology were followed and the results obtained by us are displayed in the following sections along with the heuristic behind how they have been obtained with supporting evidence.

9.1 Plot Element Detection Stage

The aim in this stage was to ensure that there was maximal overlap between the bounding boxes proposed by our object detection trained model with saved weight and the actual bounding box locations of the test image which are hidden from the model. AP aka Average precision is the score to look out for , lies between 0 - 100% , more the value of AP , higher the overlap and better the prediction made by the model . This again requires right selection of models , appropriate number of training iterations and Gamma factor . Below we display the results for various parameter settings and the incremental growth achieved by the model.



[08/27 11:11:57 d2.evaluation.coco_evaluation]: Evaluation results for bbox:

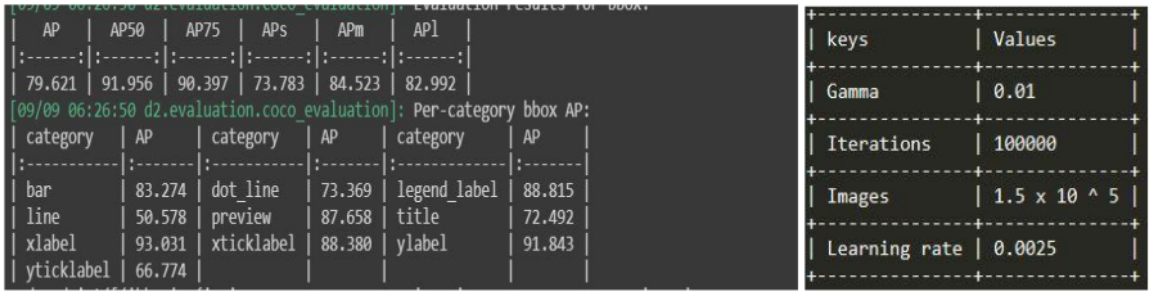
AP	AP50	AP75	APs	APm	APl
61.946	88.722	76.841	55.716	70.649	63.861

[08/27 11:11:57 d2.evaluation.coco_evaluation]: Per-category bbox AP:

category	AP	category	AP	category	AP
bar	73.387	dot_line	57.155	legend_label	74.983
line	27.329	preview	50.912	title	62.899
xlabel	76.929	xticklabel	62.948	ylabel	80.105
yticklabel	52.816				

keys	Values
Gamma	0.01
Iterations	1000
Images	50000
Learning rate	0.01

Figure 9.1 Test Accuracy After Trial - 1 With Parameter Setting on RHS



AP	AP50	AP75	APs	APm	APl
79.621	91.956	90.397	73.783	84.523	82.992

[09/09 06:26:50 d2.evaluation.coco_evaluation]: Per-category bbox AP:

category	AP	category	AP	category	AP
bar	83.274	dot_line	73.369	legend_label	88.815
line	50.578	preview	87.658	title	72.492
xlabel	93.031	xticklabel	88.380	ylabel	91.843
yticklabel	66.774				

keys	Values
Gamma	0.01
Iterations	100000
Images	1.5 x 10 ^ 5
Learning rate	0.0025

Figure 9.2 Test Accuracy After Trial - 2 With Parameter Setting on RHS

[09/16 19:01:59 d2.evaluation.coco_evaluation]: Evaluation results for bbox:

AP	AP50	AP75	APs	APm	APl
87.014	92.825	92.086	80.028	92.351	92.661

[09/16 19:01:59 d2.evaluation.coco_evaluation]: Per-category bbox AP:

category	AP	category	AP	category	AP
bar	88.876	dot_line	76.851	legend_label	95.387
line	61.344	preview	94.369	title	89.818
xlabel	97.666	xticklabel	96.352	ylabel	98.417
yticklabel	71.063				

keys	Values
Gamma	0.01
Iterations	200000
Images	1.5 x 10 ^ 5
Learning rate	0.0004

Figure 9.3 Test Accuracy After Trial - 1 With Parameter Setting on RHS

It can conclusively be seen that the AP increases linearly with the Iterations . Better training produces better results . The maximum iterations we could run was for 2 lakh iterations.Per category bbox AP value also tends to increase along with more training . This better AP helps to capture the textual and pictorial elements better in the further stages of the pipeline.

9.2 Table Question Answering Stage

The Table QA Stage outputs an answer for the question using the CSV generated from the semi-structured table generation stage. The accuracy of this stage depends on the accuracy of previous stages. The answers are categorized into two classes to arrive at the final accuracy. One type is the floating-point answers. Since the answer cannot be exact, we have allowed an error window of 5%. Values that are within this 5% range will be accepted. The other type are the String answers. Here, we consider an Exact Match metric. Answers that exactly match the ground truth values are considered towards accuracy.

We have compared our results with the PlotQA model. The PlotQA model was tested on 5860 questions from 160 images to obtain human accuracy. The paper reports the Human accuracy as 80.47%. On the DVQA dataset, the model has an accuracy of 58 % and on the PlotQA dataset, the model has an accuracy of 22.52 %. We have tested our model for a different number of images (5, 25 and 50) and the results are shown in table 9.1. The questions include open-ended, Yes/No, and structural.

Number of Images	Number of Questions	Number of Correct Questions	Accuracy (in %)
5	153	53	34.6405
25	726	293	40.3581
50	1657	619	37.3567

Table 9.1 Accuracy of Table QA model for different number of Images

CHAPTER 10

CONCLUSION AND FUTURE WORK

We have proposed an alternative solution to perform question answering on statistical charts. The input chart is passed through the PED stage to obtain the bounding box predictions of the chart. It is then passed through the OCR stage to obtain the captured text from the image. Further, the output of the OCR stage is passed through the Semi-structure table generation to obtain a table in the form of CSV. Finally, the input question and the CSV file generated is passed through the Table Question Answering stage that outputs the predicted answer. TAPAS and TABFACT models are used for the purpose of Question Answering. The PED model produced an Average Precision of 87.014 % after training for 2 lakh iterations. Further, the Table QA model was tested for a different number of images and produced significantly better results.

Our Question Answering model is restricted to answer a limited number of questions. This limitation can be addressed in the future work. The accuracy obtained by the PlotQA model and our model tested on TAPAS are significantly lower than the human performance. Hence, there is wide scope of improvement in the QA model and further research in this field.

REFERENCES / BIBLIOGRAPHY

1. Kim, Dae Hyun, Enamul Hoque, and Maneesh Agrawala. "Answering questions about charts and generating visual explanations." Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems. 2020.

2. Reddy, Revanth, et al. "Figurenet: A deep learning model for question-answering on scientific plots." *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019.

3. Sharma, Monika, et al. "ChartNet: Visual reasoning over statistical charts using MAC-Networks." 2019 International Joint Conference on Neural Networks (IJCNN). IEEE, 2019.

4. Methani, Nitesh, et al. "Plotqa: Reasoning over scientific plots." Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2020.

5. Wu, Yuxin, et al. "Detectron2. 2019." URL https://github. com/facebookresearch/detectron2 2.3 (2019).

6. Smith, Ray. "An overview of the Tesseract OCR engine." Ninth international conference on document analysis and recognition (ICDAR 2007). Vol. 2. IEEE, 2007.

7. Herzig, Jonathan, et al. "TaPas: Weakly supervised table parsing via pre-training." arXiv preprint arXiv:2004.02349 (2020).

8. Abadi, Martín, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." arXiv preprint arXiv:1603.04467 (2016).

9. Kahou, Samira Ebrahimi, et al. "Figureqa: An annotated figure dataset for visual reasoning." arXiv preprint arXiv:1710.07300 (2017).

10. Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global vectors for word representation." Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.

11. Kafle, Kushal, et al. "Dvqa: Understanding data visualizations via question answering." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

12. Berant, Jonathan, et al. "Semantic parsing on freebase from question-answer pairs." Proceedings of the 2013 conference on empirical methods in natural language processing. 2013.

APPENDIX A DEFINITIONS, ACRONYMS AND ABBREVIATIONS

Acronyms	Description
VQA	Visual Question Answering
ML	Machine Learning
NN	Neural network
CNN	Convolutional Neural network
LSTM	Long Short Term Memory
MAC	Memory Attention Composition
OCR	Optical Character Recognition
NLP	Natural Language Processing
TAPAS	TAble PArSing
OCR	Optical Character Recognition
PED	Plot Element Detection
RNN	Recurrent Neural Network
GloVe	Global Vectors for word representation
CSV	Comma Separated Values
JSON	JavaScript Object Notation

“VISUAL QUESTION ANSWERING ON STATISTICAL PLOTS”

ORIGINALITY REPORT

10%	5%	4%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to PES University Student Paper	5%
2	Nitesh Methani, Pritha Ganguly, Mitesh M. Khapra, Pratyush Kumar. "PlotQA: Reasoning over Scientific Plots", 2020 IEEE Winter Conference on Applications of Computer Vision (WACV), 2020 Publication	1%
3	Dae Hyun Kim, Enamul Hoque, Maneesh Agrawala. "Answering Questions about Charts and Generating Visual Explanations", Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, 2020 Publication	1%
4	arxiv.org Internet Source	<1%
5	www.coursehero.com Internet Source	<1%
6	venturebeat.com Internet Source	<1%

7	eprints.kfupm.edu.sa Internet Source	<1 %
8	John Mathew, Janaki Meena M. "A Survey on Object Detection from Scientific Plots", 2021 5th International Conference on Computer, Communication and Signal Processing (ICCCSP), 2021 Publication	<1 %
9	docplayer.net Internet Source	<1 %
10	krishikosh.egranth.ac.in Internet Source	<1 %
11	oro.open.ac.uk Internet Source	<1 %
12	A Lubna, Saidalavi Kalady, A. Lijiya. "MoBVQA: A Modality based Medical Image Visual Question Answering System", TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), 2019 Publication	<1 %
13	docshare.tips Internet Source	<1 %
14	par.nsf.gov Internet Source	<1 %
15	Sara Seneca, Michael A Morris, Simon Patton, Rob Elles, Jorge Sequeiros. "Experience and	<1 %

	outcome of 3 years of a European EQA scheme for genetic testing of the spinocerebellar ataxias", European Journal of Human Genetics, 2008 Publication	
16	Can Liu, Yun Han, Ruike Jiang, Xiaoru Yuan. "ADVISor: Automatic Visualization Answer for Natural-Language Question on Tabular Data", 2021 IEEE 14th Pacific Visualization Symposium (PacificVis), 2021 Publication	<1 %
17	jcheminf.biomedcentral.com Internet Source	<1 %
18	cran.r-project.org Internet Source	<1 %
19	"Computer Vision – ECCV 2018 Workshops", Springer Science and Business Media LLC, 2019 Publication	<1 %
20	open.metu.edu.tr Internet Source	<1 %
21	Tianwei Xing, Luis Garcia, Federico Cerutti, Lance Kaplan, Alun Preece, Mani Srivastava. "DeepSQA", Proceedings of the International Conference on Internet-of-Things Design and Implementation, 2021 Publication	<1 %

22	dspace.library.uvic.ca:8080 Internet Source	<1 %
23	export.arxiv.org Internet Source	<1 %
24	hdl.handle.net Internet Source	<1 %
25	hg.mozilla.org Internet Source	<1 %
26	ir.uiowa.edu Internet Source	<1 %
27	ro.scribd.com Internet Source	<1 %
28	vatraoficial.files.wordpress.com Internet Source	<1 %
29	www.ess.washington.edu Internet Source	<1 %
30	www.science.gov Internet Source	<1 %
31	Aurelie Charles. "Fairness and Wages in Mexico's Maquiladora Industry: An Empirical Analysis of Labor Demand and the Gender Wage Gap", Review of Social Economy, 2011 Publication	<1 %

Exclude quotes	On	Exclude matches	< 5 words
Exclude bibliography	On		

TapasQA - Question Answering on Statistical Plots using Google TAPAS

Himanshu Jain
Department of Computer Science
PES University, Bangalore, India
nhimanshujain@gmail.com

Sooryanath I T
Department of Computer Science
PES University, Bangalore, India
sooryanathit@gmail.com

Sneha Jayaraman
Department of Computer Science
PES University, Bangalore, India
sneha.jayraman@gmail.com

Dr. Mamatha H R
Department of Computer Science
PES University, Bangalore, India
mamathahr@pes.edu

***Abstract*—Question answering systems are used to generate answers for the questions proposed. They have been used in various applications like personal assistance and health care. This research aims to build a question answering system for statistical charts. Charts are useful in representing information in a concise and simple manner. But to answer simple questions from the charts is highly complex for a machine. We have built a state-of-the-art model that helps us address open-ended questions and Yes/No binary questions. The model consists of four stages, namely, Plot Element Detection (PED), Optical Character Recognition (OCR), semi-structured table generation and table Question-Answering (QA) stage. The input image is processed in the PED stage to generate bounding box predictions and using that the text data is recognized via the OCR stage. Various plot elements are mapped to the text data recognized to produce a semi-structured table. This table along with the questions are given to the table QA stage. Google TAble PArSing (TAPAS) is employed to answer open-ended questions and TAPAS trained on TabFact dataset is used to answer binary questions. Additional helper functions are added to these models to improve the results. PED stage is trained on Faster R-CNN and Resnet-101 models. Average Precision (AP) for different plot elements were computed after training the model for two lakh iterations. Table QA stage was tested on PlotQA dataset and evaluation was performed on 2k images per category of horizontal, vertical, line and dot plots. The results are better compared to the PlotQA model, which is the original model developed for the dataset. Employing Google Tapas has improved the results and proved to be better than other state of the art models.**

***Index Terms*—Visual Question Answering, Statistical Plots, Natural Language Processing, PlotQA, Google TAPAS**

I. Introduction

Statistical charts are an intuitive and simple way to represent data. They form a way of representing structured data in the form of graphical visualisations. Such graphical visualizations aid people in better interpreting features of data. Object detection in deep learning is a field that focuses on extricating localized datum from binary or color images. Therefore, it is useful to build a model that can localize and pick up visual data in statistical plots. It is one step towards the improvement in localized object detection capabilities. Visual plots are commonly found in research papers, scientific journals, business records e.t.c. Therefore, automation of plot analysis through the means of question-answering aids an individual to draw statistical inferences quickly from them. The most important benefit is that visual question answering models on charts will help data analysts question and understand plots on a large scale, and automate the decision-making capabilities in several sectors such as the financial sector. Given this motivation, the aim of the research is to build a Visual Question Answering system which accepts statistical plots along with questions on the plot with respect to the elements of the plot (such as intersection of the curves, area under the curve, median value and few other varieties of such relational queries) and provides answers to the questions posed. The system should discover relationships between elements of a plot and provide relational reasoning to answer questions on the plot. Given an image of a statistical plot and a corresponding question, the model must be able to generate a representation of the image, parse it into an intermediary that is well interfaced with the workline , understand the query, and generate a suitable reply. Therefore, it involves an understanding of localized image-element and the query language to be able to provide for visual reasoning. This work however restricts its scope to a certain amount of selective plots and their inner variants that are frequently occurring in most common data representations. Plots related to bar plot, line plot, and dot plots and their variants have been considered to be within the scope.

II. Previous Work

There have been attempts in the recent past to improve machine reasoning capabilities through visual question answering systems on graphical plots. The RNN architecture and the CNN-LSTM architecture form baseline comparison models with an accuracy of 75% and 60% respectively. This in comparison to human accuracy falls short by a large margin. A recent paper publication introduced the FigureNet

[1] architecture that was able to achieve an accuracy of approximately 85% on an open-source dataset. This however, only gives a yes/no binary output to a question posed, and is limited to only bar and pie charts. Another adaptation to this model, showed significant enhancements in terms of being able to answer open-ended questions on a different synthesised dataset. This domain of visual question answering on statistical plots, however, has a lot of scope of improvement in terms of future enhancements of these models. There is a possibility of expanding the types of charts to those beyond bar and pie charts or even improving on accuracy through model adaptation. In our work we made an attempt to finetune the existing work and also tried out the alternatives for table question answering stage.

III. Proposed System

A. Dataset

PlotQA dataset [2] is used for testing the TapasQA model. It consists of images of statistical plots with corresponding annotations (bounding boxes of elements of the plot) and question-answer pairs. The annotations are used to test the Plot Element Detection (PED) Stage, and the question-answer pairs are used to test the Table Question Answering (QA) Stage. The dataset splitup is shown in the Table I.

TABLE I
PlotQA Dataset Statistics

Dataset Split	#Images	#QA pairs
Train	157,070	20,249,479
Validation	33,650	4,360,648
Test	33,657	4,342,514
Total	224,377	28,952,641

B. Pipeline

In this subsection, we describe the different stages of our model to generate answers for the given input plots and questions. There are four main stages, *viz.*, (i) Plot Element Detection, (ii) Optical Character Recognition, (iii) Semi Structured Table Generation, and (iv) Table Question Answering stage. Each stage contributes towards identifying different plot elements and question structure to generate the final answer. Figure 1 shows the entire pipeline.

1) ***Plot Element Detection (PED) Stage****:* The plot image along with its annotations are passed to the object detection model - Detectron-2 for training. After training, the weights obtained are used for obtaining the bounding box around the plot elements.

Detectron-2 [3] is faster, flexible and vast in terms of configuration, models and implementation due to its API availability when compared to its parent (Detectron). Thus, we have used it as the object detection and bounding box generation tool. This stage consists of a deep learning FASTER R-CNN module for object localization, because our main aim is to localize and produce bounding boxes around the plot elements and extricate them rather than classifying an image. The purpose of choosing Faster-RCNN as our object detection model is due to the presence of the RPN (Regional proposal network is known for locating feature targets accurately) and the inference time. This module produces the bounding box annotation of all involved plot elements captured by the object detection model which is fine-tuned and trained. Bounding boxes are a vector representation of the plot elements in a graph, which refer to the coordinates of the object (top_x , top_y , bottom_x , bottom_y); We effectively need only these four points to draw a bounding box around a plot element. Bounding box values of the plot elements like x-label, title, y-lable, the bar, the line, the dot are all obtained. Resnet-101 [4] is used as our feature extractor due to skip connections and residual blocks which can reduce the problem of vanishing gradients in a very deep networks. The output of a model will be a JSON file which maps all the detected objects (plot elements) to the class to which it belongs, with an appropriate confidence value and its bounding box tensor. There are a total of 11 classes that have been defined by us. These are divided into textual and visual elements. The title of the plot, x-label, y-label, x-tick label, y-tick label and the legend label form the textual elements. Bar, line, dot-line and legend preview form the visual elements. Additionally, the background of the plot forms an element. Using the JSON file, we generate a more readable and easily understandable textually formatted data that can be used for further processing in the pipeline.

2) ***Optical Character Recognition (OCR) Stage****:* There are a total of 10 different plot elements that can be found in any statistical plot. These can be grouped into two categories: Textual Elements and Visual Elements. Textual elements correspond to the title of the plot, y-axis label, x-axis label, x-tick, y-tick values and the labels corresponding to the legend. To read the textual and numeric data off the textual components, we make use of the formatted textual output from the previous stage, and an Optical Character Recognition module [5]. With the help of bounding box coordinates extracted, we can accurately and locally capture the text information within the bounding boxes rather than passing an entire image into the OCR module. The captured text is then read using OCR and classified into its category. The OCR module used is pyocr which is a wrapper for the Tesseract OCR engine. Detected textual elements are cropped to the bounding box size (which is obtained from the previous stage), then converted to gray-scale and passed onto the pyocr module. The output of this stage is textual data corresponding to the detected textual elements.

3) ***Semi Structured Table Generation Stage****:* The simple bar graph goes through the plot element detection stage, and a text file emerges as the outcome as discussed in the previous stages. This text file will then be passed onto the OCR stage along with the image to extract the textual content within the boxed coordinates. Following which the information is structured in the semi structured table generation phase. The

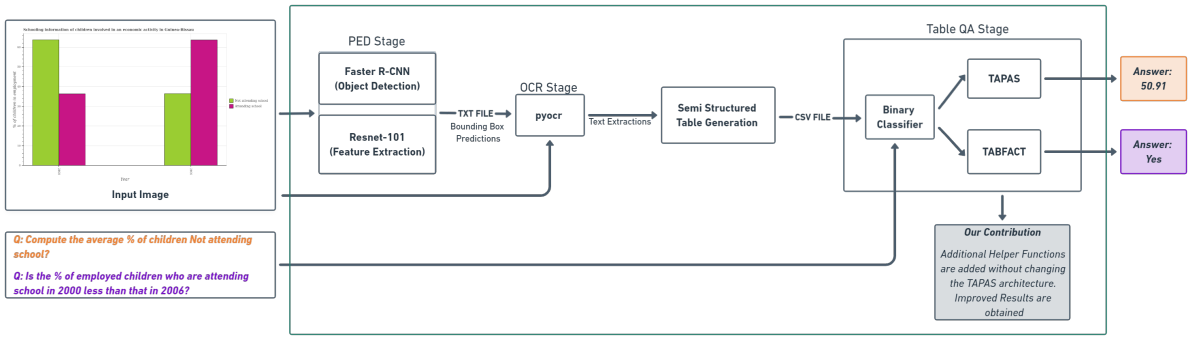


Fig. 1. Tapas-QA Pipeline

output of this stage is a semi-structured table (CSV file) that encapsulates all of the data in the statistical plot. For the textual elements, we have already obtained textual data. This stage is responsible for mapping legend values to the legend color, x-ticks to the x-axis label and the y-ticks to the y-axis label. This is done by associating the legend / x-tick / y-tick value bounding box to the closest legend color / x-axis / y-axis boundary respectively. For the visual elements, each element is associated with an axis, and a corresponding legend. The color of the visual element is matched with the legend colors, and the legend of the closest match is associated with the element. To find the value associated with the bar, the information of height is taken from the bounding box representation, and the closest y-tick is mapped. Doing this for all visual elements will fill the table and result in a table that is stored as a comma separated file. This file is then passed to the Table QA stage

*4) **Table Question Answering (QA) Stage**:* Given a semi-structured table and a relevant natural language question as input, this stage is responsible for producing an answer to the question from the table as an output. The questions can be classified into two types. The first type corresponds to open-ended questions that have an unrestricted answer domain. The second type corresponds to questions that require a Yes/No (binary) answer. To handle open-ended questions, we have made use of the existing TaPas (Table Parsing) model [6]. This model is based on the BERT's encoder with certain modifications. Positional embeddings are used to encode tabular data, and two additional classification layers are introduced to select cells of the table and the aggregation operation to be performed. Our work makes use of a pre-trained TaPas model that has been trained on the WikiTables Questions dataset [7] with intermediate pre-training. This model can handle three types of aggregation operations - SUM, COUNT, AVERAGE. To add to the capabilities of this model, we have added other operations such as RATIO, DIFFERENCE, MEDIAN, TREND, RANGE and QUARTILES. To handle questions that require a Yes/No answer, we have used a TaPas model trained on the TabFact dataset [8]. This is a dataset used for table entailment and fact verification. We have extended its capabilities by adding other operations like in the earlier mentioned model. An important aspect here is that given an input question we would need to know what type of question it is (whether it is an open-domain question or a yes/no question). For this, we have implemented a binary question classifier.

*5) **Binary Classifier**:* There are two categories of questions addressed: Open-ended and Yes/No. Each of this addressed by an independent model and hence to integrate into a single pipeline, a binary classifier is used. Binary classifier model classifies the given input question into Yes/No class (class 0) or Open-ended class (class 1) A dataset is prepared for this purpose from the PlotQA data. The model is trained on questions of all types of plots (i,e, vbar_categorical, hbar_categorical, dot_line, line) and the answers which are converted into the categories of 0 or 1. The Deep Learning model is trained to classify the input questions into correct classes. Figure 2 shows the architecture of the binary classifier. The model is trained using the GloVe embeddings [9] and binary cross-entropy with adam optimizer is employed. The model was trained for 10 Epochs in a batch size of 10. The trained model was saved and loaded to classify test images. If the model outputs class 0, the question is passed to TABFACT model and if the model outputs class 1, the question is passed to TAPAS model

IV. Experiments

A. Training Details

Plot Element Detection Stage: For PED, we have trained a Faster R-CNN object detection model on the Resnet-101 feature extractor. The model has been trained on Train split of the dataset with a batch size of 512 for 200,000 iterations. The initial learning rate has been kept to 0.004.

Binary Classifier: As mentioned earlier, the binary classifier is trained using the GloVe embeddings and binary cross-entropy with the adam optimizer. The model was trained for 10 Epochs with a batch size of 10.

B. Evaluation Metric

Plot Element Detection Stage: We have used Average Precision as the evaluation metric. The higher the value of AP, the higher the overlap and better is the prediction made by

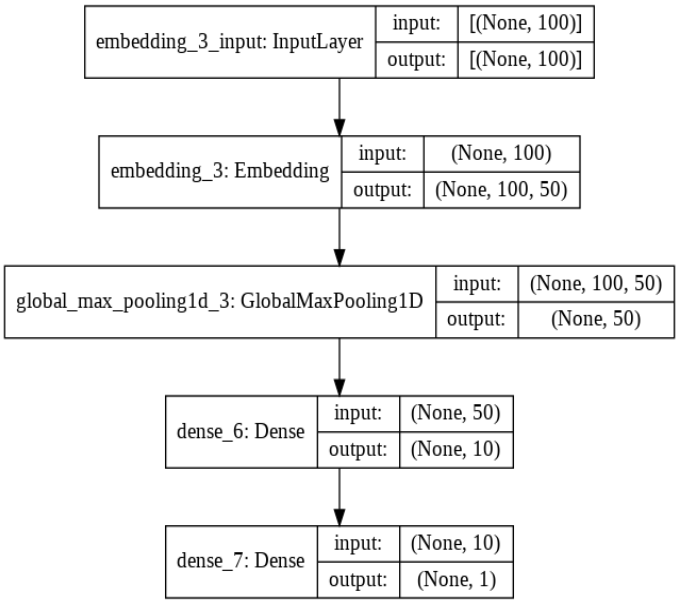


Fig. 2. Binary Classifier

the model. This again requires the right selection of models, appropriate number of training iterations and learning rate. Below we display the results for various parameter settings and the incremental growth achieved by the model.

TABLE II
AVERAGE PRECISION AT DIFFERENT NUMBER OF ITERATIONS

Number of Iterations	AP	AP50	AP75
100,000	79.621	91.956	90.397
130,000	86.584	92.819	92.076
150,000	87.014	92.825	92.086
170,000	87.053	92.823	92.083
200,000	87.179	92.823	92.088

It can conclusively be seen that the AP increases with the number of iterations. Better training produces better results. The total number of iterations we have trained for is 200,000. Per category bounding box AP value also tends to increase along with more training. Higher value of AP helps us to capture the textual and pictorial elements better in the further stages of the pipeline.

Table Question Answering (QA) Stage: We have used accuracy as the evaluation metric. For textual answers, the answer would contribute to the accuracy only if an exact match was found between the expected and the predicted answers. However, in the case of numeric answers, we have allowed for an error window of 5%. Answer values within this range will be considered correct.

V. OBSERVATIONS AND RESULTS

A. Plot Element Detection Stage

The aim in this stage was to ensure that there was maximal overlap between the bounding boxes proposed by our object detection trained model with saved weight and the actual bounding box locations of the test image which are hidden from the model. AP aka Average precision is the score to look out for. Below is the class wise split-up of Average Precision obtained on the model trained on 200,000 iterations.

TABLE III
PED EVALUATION

Class	AP
Bar	88.819
Line	61.466
Dot-Line	77.439
X-Label	97.840
Y-Label	98.438
Title	90.056
X-tick Label	96.500
Y-tick Label	71.094
Legend Label	95.550
Preview	94.589

B. Table Question Answering (QA) Stage

The Table QA Stage outputs an answer for the question using the CSV generated from the semi-structured table generation stage. The accuracy of this stage depends on the accuracy of previous stages. The answers are categorized into two classes to arrive at the final accuracy. One type is the floating-point answers. Since the answer cannot be exact, we have allowed an error window of 5%. Values that are within this 5% range will be accepted. The other type are the String answers. Here, we consider an Exact Match metric. Answers that exactly match the ground truth values are considered towards accuracy.

TABLE IV
TABLE QA RESULTS

Plot Type	Number of Images Tested	Total Number of Questions	Number of Correct Answers	Average Accuracy (in %)
Dot	2000	53970	25104	46.965499
Vertical	2000	47940	19898	41.474200
Horizontal	2000	49241	20128	40.990114
Line	2000	35353	14077	36.669402

We have compared our results with the PlotQA model. The PlotQA model was tested on 5860 questions from 160 images to obtain human accuracy. The PlotQA paper reports the Human accuracy as 80.47%, an accuracy of 58% on the DVQA dataset, and an accuracy of 22.52% on the PlotQA dataset. Our model has an average accuracy of 41.52% across all types of plots and structural, data-retreival and reasoning type questions.

VI. CONCLUSION

We have proposed an alternative solution to perform question answering on statistical charts. The input chart is passed through the PED stage to obtain the bounding box predictions of the chart. It is then passed through the OCR stage to obtain the captured text from the image. Further, the output of the OCR stage is passed through the Semi-structure table

generation to obtain a table in the form of CSV. Finally, the input question and the CSV file generated is passed through the Table Question Answering stage that outputs the predicted answer. TAPAS and TABFACT models are used for the purpose of Question Answering. The PED model produced an Average Precision of 87.014 % after training for 2 lakh iterations. Further, the Table QA model was tested for a different number of images and produced significantly better results.

Our Question Answering model is restricted to answer a limited number of questions. This limitation can be addressed in the future work. The accuracy obtained by the PlotQA model and our model tested on TAPAS are significantly lower than the human performance. Hence, there is wide scope of improvement in the QA model and further research in this field.

REFERENCES

[1] R. Reddy, R. Ramesh, A. Deshpande, and M. M. Khapra, "Figurenet: A deep learning model for question-answering on scientific plots," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[2] N. Methani, P. Ganguly, M. M. Khapra, and P. Kumar, "Plotqa: Reasoning over scientific plots," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 1527–1536.

[3] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[5] R. Smith, "An overview of the tesseract ocr engine," in *Ninth international conference on document analysis and recognition (ICDAR 2007)*, vol. 2. IEEE, 2007, pp. 629–633.

[6] J. Herzig, P. K. Nowak, T. Müller, F. Piccinno, and J. M. Eisenschlos, "Tapas: Weakly supervised table parsing via pre-training," *arXiv preprint arXiv:2004.02349*, 2020.

[7] P. Pasupat and P. Liang, "Compositional semantic parsing on semi-structured tables," *arXiv preprint arXiv:1508.00305*, 2015.

[8] J. C. Y. Z. H. W. S. L. X. Z. Wenhu Chen, Hongmin Wang and W. Y. Wang, "Tabfact : A large-scale dataset for table-based fact verification," in *International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia, April 2020.

[9] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

APPENDIX A
QUESTION TEMPLATE

1) Difference
   - Keyword = "difference between"
   - The two entities must be separated by "and"
2) Median
   - Keyword = "median"
   - Column name for which the median has to be found
3) Ratio
   - Keyword = "Ratio"
   - QUANTITY_1 "to" QUANTITY_2
4) Trend
   - Keyword = "trend"
   - List of comma separated entries followed by "in" COL_NAME

APPENDIX B
FULL EXAMPLE

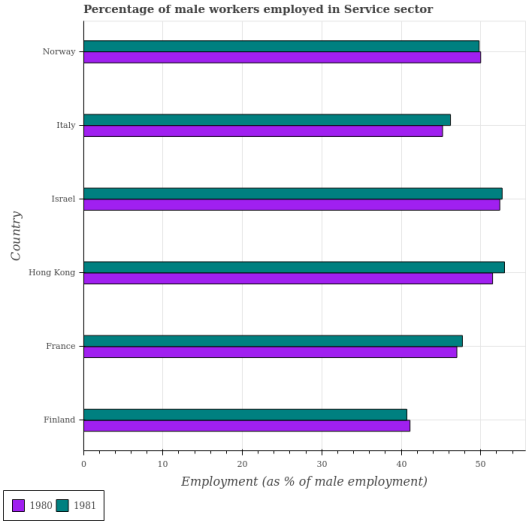


Fig. 3. Grouped Horizontal Bar Chart

	Country	1980	1981
0	Israel	54.191570	54.512780
1	France	48.970106	48.401403
2	Finland	41.569151	41.217884
3	Norway	51.488818	51.018932
4	Italy	46.373521	46.332053
5	Hong Kong	53.624025	54.126604

Fig. 4. CSV Output

Types of questions addressed:

1) Count
   Q: what are the total number of countries
   A: 6

2) Sum
   Q: what is the total number of male workers employed in 1980
   A: 296.2171903097152

3) Average
   Q: what is the average number of male workers in the year 1980
   A: 49.36953171828586

   Q: what is the average number of male workers in the year 1981
   A: 49.26827598218275

4) Minimum
   Q: across all countries, what is the minimum percentage of male workers employed in service sector in 1980
   A: 41.56915064054187

5) Maximum
   Q: across all countries, what is the maximum percentage of male workers employed in service sector in 1980
   A: 54.191569798758344

6) Difference
   Q: what is the difference between the average number of male workers employed for the year 1980 and 1981
   A: DIFFERENCE = 0.10125573610311278

   Q: what is the difference between the number of male worker employed for the country france in 1980 and 1981
   A: DIFFERENCE = 0.5687026318456105

7) Median
   Q: what is the median number of male workers employed in the year 1980
   A: MEDIAN = 50.22946206732409

8) Ratio
   Q: what is the ratio of male workers employed in 1981 to 1980 for the country hong kong
   A: RATIO = 1.0093722657385955

9) Trend (Increasing or Decreasing) Q: what is the trend of male workers employed for the countries finland, france, hong kong in 1980
   A: TREND = INCREASING

   Q: what is the trend of male workers employed for the countries israel, italy in 1980
   A: TREND = DECREASING

   Q: what is the trend of male workers employed for the countries israel, italy, norway in 1980
   A: TREND = NONE

10) Selection operation on cell
    Q: what is the number of male workers employed for country france in the year 1981
    A: 48.40140328049439

11) Select operation on cell after applying aggregation operation
    Q: which country has the minimum number of male workers employed in the year 1981
    A: Finland

12) Selection and Aggregation operation on subset of rows
    Q: what is the sum of male workers employed for the countries france, finland in the year 1980
    A: 90.53925655288188

    Q: what is the average number of male workers employed for the countries france, finland in the year 1980
    A: 45.26962827644094

    Q: what is the maximum number of male workers employed for the countries france, finland in the year 1980
    A: 48.97010591234

13) Project operation on column
    Q: what are the names all the countries
    A: Finland, France, Hong Kong, Israel, Italy, Norway

    Q: list out the countries
    A: Finland, France, Hong Kong, Israel, Italy, Norway

14) Range
    Q: what is the range of % of male employment for the year 1980
    A: RANGE = 12.622419158216474

15) Quartiles (Q1 and Q3)
    Q: find the quartiles for the year 1980
    A: FIRST QUARTILE (Q1) = 43.97133559037385
    SECOND QUARTILE (Q2) = 50.22946206732409
    THIRD QUARTILE (Q3) = 53.90779749715967

16) IQR
    Q: find the interquartile range for the year 1980
    A: INTER-QUARTILE RANGE = 9.936461906785823

17) Structural Query
    Q: what is the title of the graph ?

A: TITLE OF THE GRAPH = Percentage of male workers employed in Service sector

Q: what is the label or title of the x-axis ?
A: X-LABEL = Employment (03 % of male employment)

Q: what is the label or title of the y-axis ?
A: Y-LABEL = Country

18) YES / NO
Q: in the year 1981, hong kong has the highest employment percentage.
A: YES

Q: for the year 1980, italy has a lower employment percentage than norway.
A: YES

Q: the average employment rate of hong kong is greater than the average employment rate of finland.
A: YES

Q: is the percentage of male workers employed in service sector in 1980 in italy less than that in norway ?
A: YES