Program Synthesis for Complex QA on Charts via Probabilistic Grammar Based Filtered Iterative Back-Translation

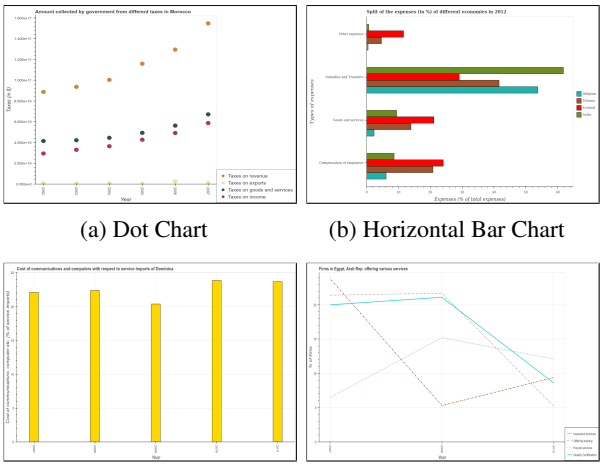
Shabbirhussain Bhaisaheb, Shubham Paliwal, Rajaswa Patil,
Manasi Patwardhan, Lovekesh Vig, Gautam Shroff
TCS Research, India
{shabbirhussain.b, shubham.p3, patil.rajaswa,
manasi.patwardhan, lovekesh.vig, gautam.shroff}@tcs.com

Abstract

Answering complex reasoning questions from chart images is a challenging problem requiring a combination of natural language understanding, fine-grained perception, and analytical reasoning. Current chart based Question Answering (QA) approaches largely address structural, visual or simple data retrieval type questions with fixed-vocabulary answers and perform poorly on reasoning queries. We focus on answering realistic, complex, reasoning-based questions where the answer needs to be computed and not selected from a fixed set of choices. Our approach employs a neural semantic parser to transform Natural Language (NL) questions into SQL programs and execute them on a standardized schema populated from the extracted chart contents. In the absence of program annotations, i.e., in a weak supervision setting, we obtain initial SQL predictions from a pre-trained CodeT5 semantic parser and employ Filtered Iterative Back-Translation (FIBT) for iteratively augmenting our NL-SQL training set. The forward (neural semantic parser) and backward (language model) models are initially trained with an external NL-SQL bootstrapping data. We iteratively move towards the required NL query distribution by generating NL questions from the synthesized SQL programs using a Probabilistic Context-Free Grammar (PCFG) where the production rule probabilities are induced to be inversely proportional to the probabilities in the training data. We filter out the generated NL queries with mismatched structure and compositions. Our FIBT approach achieves State-of-the-Art (SOTA) results on reasoning-based queries in the PlotQA dataset yielding a test accuracy of 60.44%, superseding the previous baselines by a large margin.

1 Introduction

Charts and plots are compact visualization techniques capturing illustrated facts that are frequently used in scientific and financial documents for summarizing observations and drawing conclusions about the underlying data. Inferring relevant conclusions from charts entails answering complex reasoning style queries, a task which has so far proved challenging to automate. Most existing approaches and datasets for automatic QA over charts specifically focus on structural, visual, relational or simple data retrieval type queries (Chaudhry et al., 2020; Siegel et al., 2016; Kim et al., 2020). Also, as depicted in Table 1 many approaches assume either binary answers or assume the answer belongs to a fixed vocabulary.



(a) Dot Chart (b) Horizontal Bar Chart

(c) Vertical Bar Chart (d) Line Chart

Figure 1: PlotQA Chart Types

On the other hand, for real world applications, more complex reasoning based questions have to be answered which involves a combination of perception, language understanding, and reasoning. For example, to answer the question depicted in Table 1, for the chart (d) in Figure 1, the following steps have to be followed: (i) Refer to the legend to infer that the 'percentage of firms offering quality certification' are depicted by the solid sky blue line, (ii) For each year value on the X-axis, retrieve the corresponding Y-values depicted by the solid sky blue line, (iii) If the value of corresponding year y is $V(y)$, then find if $\forall y_i$ and y_j, where $i \neq j$,

Findings of the Association for Computational Linguistics: EACL 2023, pages 2501–2515
May 2-6, 2023

if $y_i > y_j$ then $V(y_i) >= V(y_j)$, to determine if the values are monotonically increasing. In this work, we address such complex reasoning style questions on charts. These questions may also involve nested arithmetic and aggregation operations over the chart data and thus the answer is not necessarily derived from a fixed vocabulary, or extracted from the chart text. We evaluate our approach against the reasoning questions provided in the PlotQA V2 dataset (Methani et al., 2020).

We define a common schema for the data across all chart types and employ a state-of-the-art chart visual extractor to populate the schema with chart data. The PlotQA dataset does not provide any SQL program annotations for the Natural Language (NL) questions (only answers). We automatically generate SQL programs for these NL questions by using a Filtered Iterative Back-Translation (FIBT) approach (Hoang et al., 2018) and execute these programs on the extracted schema to compute the answers. We use the SPIDER (NL to SQL) dataset to train both the forward and backward models for FIBT. The NL query distribution of this dataset is different from the required PlotQA query distribution, in terms of query composition, schema (database) structure and chart domains. We build on the observations of (Guo et al., 2020), who empirically show that Iterative Back Translation (IBT) improves the performance of compositional generalization while generating logical forms from NL questions by correcting errors in the pseudo-parallel data at each iteration.

We define a Probabilistic Context-Free Grammar (PCFG), as a subset of the SQL grammar, to sample SQL programs executable on the extracted chart schema. Existing PCFG based data augmentation approaches for semantic parsing (Wang et al., 2021b), synthesize SQLs following similar compositions to that of a given query set by inducing proportional grammar probabilities. Our approach differs from these as we set the probabilities of our PCFG to be inversely proportional to the set-of programs in the training data. This results in synthesis of SQL programs that (i) were not present in the current training data and (ii) follow the program distribution of the SQL programs needed to compute the answers for the PlotQA questions. We iteratively augment the training data by generating NL questions from SQL programs. We further use denotations and a novel compositional similarity based filtration strategy for removing noisy NL-SQL pairs. We observe that this data augmentation and filtration strategy results in improvement in the PlotQA execution accuracy for every iteration of FIBT, finally achieving State-of-the-Art performance on the reasoning-based queries for the PlotQA dataset with a 60.44% test set accuracy, superseding the previous baseline (14.82%) (Methani et al., 2020) by a large marginand even surpassing human performance. The ChartQA model(Masry et al., 2022) showed that the T5 language model offers the best overall performance for the questions in PlotQA. We demonstrate that our approach surpasses T5 for complex reasoning type of questions in PlotQA with OOV answers. The main contributions of this work are:

- To the best of our knowledge, ours is the first approach to effectively address reasoning style NL questions over charts whose answers are computed and not restricted to a fixed vocabulary.
- In absence of program annotations, we propose a weakly supervised FIBT approach for SQL synthesis with novel data augmentation and filtration strategies to adapt the Neural Parser to more closely follow target NL-Question distribution.
- Our approach allows us to achieve State-of-the-Art results on PlotQA reasoning-based questions with a 60.44% test set accuracy, superseding the previous baseline (14.82%) by a large marginand even surpassing human performance (58.70%).
- As opposed to existing end-to-end approaches (Singh and Shekhar, 2020; Kafle et al., 2020; Chaudhry et al., 2020) , our approach is more interpretable as we can track reasoning patterns in the synthesized programs via the generated programs.

2 Related Work

2.1 Datasets for Chart Q&A

Chart QA datasets such as DVQA (Kafle et al., 2018) or FVQA (Kahou et al., 2017) are synthetically generated with limited variations, containing simple binary or fixed-vocabulary questions. To avoid these biases, Leaf-QA (Chaudhry et al., 2020) and PlotQA (Methani et al., 2020) datasets are constructed from open real-world sources from World Bank, Government, Global Terrorism Database, etc.Questions in these datasets are in English and are templatized but paraphrased to prevent models

NL Questions	(a) Dot Chart	(b) Horizontal Bar	(c) Vertical Bar	(d) Line Chart
Structural	Is the number of dotlines equal to the number of legend labels? (Y/N)	How many groups of bars are there? (Fixed)	Does the graph contain any zero values? (Y/N)	Does the graph contain grids? (Y/N)
Retrieval	What is the amount collected as tax on revenue in 2005? (Open)	What is the label of the 4th group of bars from the top? (Chart)	What is the label or title of the Y-axis? (Chart)	What is the title of the graph? (Chart)
Reasoning	Is the difference between the amount collected as tax on goods in 2003 and 2007 greater than the difference between the amount collected as tax on exports in 2003 and 2007? (Y/N)	What is the difference between the highest and the second highest percentage of amount spent on other expenses? (Open)	Do a majority of the years between 2008 and 2011 (inclusive) have cost of communications and computer greater than 10? (Y/N)	Does the percentage of firms offering quality certification monotonically increase over the years? (Y/N)

Table 1: PlotQA Questions for Charts in Figure 1, Answer Types: Yes/No, Fixed, Chart or Open Vocabulary

Data	Images		Questions		
Split	Total	R*	Total	Reasoning	R*
Train	157,070	12,934	20,249,479	16,593,656	69,000
Valid	33,650	3,110	4,360,648	3,574,081	13,740
Test	33,657	-	4,342,514	3,559,392	-
Total	224,377	16,044	28,952,641	23,727,129	82,740

Table 2: PlotQA V2 Dataset Statistics. R*: Representative images (c_{rep}) and questions (q_{rep}) used for FIBT

from memorizing the templates. Both datasets have a significant proportion of analytical reasoning queries, however the PlotQA dataset has 81.95% complex value-based queries, requiring stronger numerical and analytical reasoning capabilities (Chaudhry et al., 2020). Also 80.84% PlotQA queries have answers from an open vocabulary. Moreover, ∼82% of questions in PlotQA are reasoning based, as opposed to the recently introduced CharQA dataset (Masry et al., 2022), which has only 43% (compositional) reasoning based questions. Since our main focus is on answering complex questions requiring numerical and analytical reasoning, we use the reasoning based questions in the extended version (V2) of the publicly available PlotQA dataset (Table 2) to evaluate our approach.

2.2 Chart Q&A Approaches

Existing end-to-end approaches use deep models to combine image and question features at various levels of granularity (Kafle et al., 2020). A recent approach(Chaudhry et al., 2020) fuses the chart entities extracted using a Masked RCNN and the NL question using spatial attention to predict the answer. (Singh and Shekhar, 2020) use a structural transformer-based learning that takes the question encoding as input and uses the feature maps of the chart's visual elements, with its localization information used as positional encodings. These approaches provide results on previously mentioned FVQA, DVQA, and LeafQA datasets on relatively simpler queries. Recently, (Masry et al., 2022; Masry and Hoque, 2021) published benchmarks for chart QA using several end-to-end approaches including, VL-T5 (Cho et al., 2021), TAPAS (Herzig et al., 2020) and VisionTaPas, which is an extension of TAPAS (Masry et al., 2022) and T5 (Raffel et al., 2019). TAPAS is able to address very simple aggregation type queries and cannot handle complex queries with nested aggregation and arithmetic operations and thus provides poor results on PlotQA (12.90%). T5 provides the best reported results for PlotQA V2 (56.22% test accuracy for all queries). Our proposed approach, designed for complex reasoning type of queries, surpasses their results. Unlike prior end-to-end approaches, we adopt a two staged approach, which not only provides us SOTA results, but allows for more interpretability. Along similar lines, (Methani et al., 2020; Kim et al., 2020) propose a multi-stage solution, where the chart extractions are stored in a semi-structured form, and pre-defined rule-based semantic parsing (Pasupat and Liang, 2015) converts the queries into a logical form. However, these approaches do not generalize to queries not expressible by the grammar rules defined for other datasets, leading to very low test accuracy especially for complex reasoning type of queries (14.82% for PlotQA). An elaborate listing of prior work on Table Q&A and Semantic Parsing and the comparison with our approach highlighting our novelty is in Appendix A.

3 Problem Definition

Our task is defined as follows: given a chart c and a question q on the chart, output a value a that answers the question according to the information represented in the chart. The system has access to a training set $D_{chart} = (q_{chart}, c_{chart}, a_{chart})_1^N$ of questions, charts, and answers. The charts and corresponding questions in test data do not appear during training. We assume availability of a bootstrapping dataset of $D_{tr} = (s_{tr}, q_{tr}, p_{tr})_1^M$, where s are the database schema, q are the Natural Language (NL) queries posed on the schema and p are the SQL programs corresponding to the NL queries. The domain of the charts c_{chart} can be distinct from the domain of the schema s_{tr}.

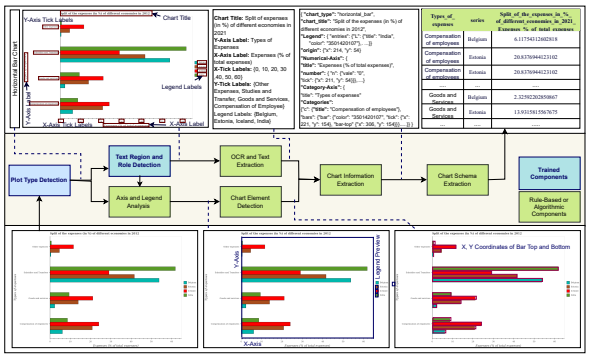


Figure 2: Chart Schema Extraction Vision Pipeline

We assume that the SQL programs p_{tr} share the primitive arithmetic, aggregation and logical operations (SUM, DIFFERENCE, RATIO, AVERAGE, MEDIAN, MAXIMUM, MINIMUM, GREATER THAN, LESS THAN) with the SQL programs required to answer the NL questions q_{chart}. However, distributions over the primitive operations and their compositions can differ.

4 Approach

We follow a two stage approach. In the first stage, we use a computer vision pipeline to extract chart and store the chart data in a format (schema) s_{chart}, common across all the chart types. In the second stage, we synthesize SQL programs for the NL questions in the dataset by using Filtered Iterative Back-Translation (FIBT). We use SQL as the target logical form for program synthesis because it's grammar (i) is well suited to the tabular structure of our extracted data and (ii) includes the primitive operations required to be handled for the NL queries in PlotQA. Also, SQL is close to NL and easy to understand, allowing for interpretability.

4.1 Chart Schema Extraction

In the first stage, we extract chart information using the following pipeline: (i) Chart Type Detection (Trained) (ii) Text Region and Role Detection (Trained) (iii) OCR and Text Extraction (Algorithm) (iv) Axis and Legend Extraction (Rule-Based) (iv) OCR and Text Extraction (Algorithm) (v) Chart Information extraction (Algorithm).

Since the chart types have distinct visual features we fine-tune a Resnet-34 model pretrained on ImageNet to detect chart types, using chart type labels provided by PlotQA. The text present in the image is detected by employing the CRAFT model (Baek et al., 2019). However, CRAFT frequently misses isolated characters and often yields partial detection of text regions. We propose an approach which corrects partially detected text, segments out the corrected text region and identifies text-role labels (such as chart title, legend labels, X/Y-axis, and X/Y-tick labels) for the text regions (Appendix B). We use Tesseract 4.0.0 (Smith, 2007) to extract text from the detected regions and tag them to the corresponding roles. The results of the chart element extractions and text role region extractions are described in Appendix F. We define rules to identify (i) the origin, axes and chart region from the detected chart lines by a line detection algorithm (Paliwal et al., 2021), (ii) location of the legend previews and their styles (color and pattern) using the detected legend-labels, and (iii) chart elements (bars, dots, lines) which are regions matching with each legend preview style. We extract a schema (table) from the above available chart information, by filtering noise (Appendix C) and extracting the data series elements (Appendix E).

Henceforth, we use the following nomenclature. For the horizontal-bar charts illustrated in Figure 1 (b), we refer to the X-axis as the Numerical-axis and Y-axis as the Categorical-axis. For the remaining chart types in Figure 1 viz. dot, vertical bar, and line chart, the nomenclature is reversed. We call each legend label as a series. Thus, the extracted chart information is in the form of a set-of tuples <$category\_label$, $series\_label$, $numerical\_value$>, with the schema (table) header being category axis label, series, and a string formed by concatenating the chart title with the numerical axis label. We store these tables s_{chart} in the SQLite3 database to facilitate the execution of synthesized SQL programs (Section 4.2) on the schema. As 'Median' is not an in-built aggregation operation for SQLite3, we define a stored procedure for the same.

4.2 SQL Program Synthesis

As the part of the second stage, we execute Filtered Iterative Back Translation (FIBT) (Algorithm 1), to train the neural semantic parser $M_{NL->P}$, which is used to synthesize SQL programs for the reasoning questions in the PlotQA test set. The generated SQLs are executed on the test set chart schema s_{chart}, extracted from the corresponding chart image c_{chart}, to compute the final answer. This answer is compared with the ground truth answer a_{chart} to calculate the test accuracy.

4.2.1 Bootstrapping Data

We use SPIDER (Yu et al., 2018) augmented with a few (359) NL-SQL query pairs as the bootstrapping dataset D_{tr} to initialize the parameters of the forward $M_{NL->P}$ and the backward $M_{P->NL}$ models of FIBT. The augmented query pairs are defined to include the primitive operations (DIFFERENCE, RATIO, LESS THAN and MEDIAN), required by the PlotQA NL questions q_{chart} but missing in the SPIDER SQLs, leading to the bootstrapping data q_{tr} and the PlotQA questions q_{chart} sharing the same set-of primitive operations. Inclusion of such query pairs allows the models to learn these primitive operations (followed by their compositions in the subsequent FIBT iterations), which otherwise would not be possible.The query pairs are synthesized using templates. For example, the templates used to synthesize NL-SQL pairs for RATIO operation is: NL: "What is the ratio of the *numerical column name* having *categorical column* value a *x* to that of *categorical column* value of *y* ?", SQL:

```
"SELECT T1.numerical_column_name /
T2.numerical_column_name FROM
table_name T1, table_name T2 WHERE
T1.categorical_column_name =
'categorical_column_value_x' AND
T2.categorical_column_name =
'categorical_column_value_y'"
```

These queries are synthesized on a subset of SPIDER schema tables, whose structure match with our extracted chart schema. Thus in the above template, the 'numerical_column_name' is the name of a column of a table belonging to one of the SPIDER database schemas, having a numerical datatype. and 'categorical_column_name' is the column name of the same table, with text datatype with values x and y as its entries.

4.2.2 Probabilistic Context Free Grammar

We define Probabilistic Context Free Grammar (PCFG) depicted in Table 7 in the Appendix, as a subset of the SQL grammar to synthesize SQL programs: (i) to address possible compositions of primitive operations required for the PlotQA NL questions q_{chart} and (ii) executable on the schema s_{chart}. To synthesize SQL programs whose distribution match with the programs for the NL questions in the PlotQA, but are not covered by the current training data D_{tr}, we induce the probability (P_{inv}) for each of the production rules R in the PCFG with the heuristics depicted in Equation 1.

$$P_{inv}(R) = \frac{Wt(R)}{\sum_{RHS(r)=RHS(R)} Wt(r)} \quad (1)$$

$$Wt(R) = \left(\frac{MAX_{RHS(r)=RHS(R)}(P(r))}{P(R)}\right)^{1-\alpha} \quad (2)$$

$P(R)$ gives the probability with which a rule R is triggered by the set-of SQL queries existing in the training data D_{tr}. $RHS(R)$ is the Right Hand Side of the production rule R. Thus, $RHS(r) = RHS(R)$ provides the set of all production rules which share the source node (RHS) with the rule R. α is the hyper-parameter controlling the skewedness of the distribution over the prodcution rules. Lower the value more skewed is the distribution. For our experiments α = 0.8.

Algorithm 1: FIBT

Input : $D_{chart} = \{q_{chart}, s_{chart}, a_{chart}\}_1^N$, Defined PCFG
Output : Trained Semantic Parser $M_{NL->P}$
Initial Stage : Bootstrapping data $D_{tr} = \{s_{tr}, q_{tr}, p_{tr}\}_1^M$
q_{rep} = ***sample(cluster(generalize(*** q_{chart}***)))*** where $q_{rep} \subset q_{chart}$
$D_{rep} = (q_{rep}, s_{rep}, a_{rep})_1^n, n << N$
p_{filter} = Φ

while $M_{NL->P}$ *and* $M_{P->NL}$ *have not converged* **do**
  Train $M_{NL->P}$ on D_{tr} `// Forward Pass`
  Feed q_{rep} to $M_{NL->P}$ to generate p_{rep}
  Execute p_{rep} on schema s_{rep} to compute a_c
  if a_c == a_{rep} `// Filter`
  then
    Add $(s_{rep}, q_{rep}, p_{rep})$ to D_{tr} Remove $(s_{rep}, q_{rep}, p_{rep})$ from D_{rep}
  end if
  Train $M_{P->NL}$ on D_{tr} `// Backward Pass`
  induce(PCFG , (p_{tr} + p_{filter}) ***)*** (Equation 1)
  Sample SQL p_{syn} on s_{chart} from PCFG.
  Feed p_{syn} to $M_{P->NL}$ to generate q_{synth}
  filter_flag = 1
  if ***max_sim(generalize(*** q_{synth} ***))*** , ***generalize(*** q_{rep} ***)*** > *threshold* `// Filter1`
  then
    if ***look_up(*** q_{synth}, p_{synth} ***)*** `// Filter2`
    then
      Add $(s_{chart}, q_{synth}, p_{synth})$ to D_{tr} `// Augment`
      filter_flag = 0
    end if
  end if
  if *filter_flag == 1* **then**
    Add p_{synth} to p_{filter}
  end if
end while

4.2.3 Sampling Representative Questions

As depicted in Table 2, PlotQA has ∼16.6 Million reasoning based NL questions as the part of the training data. For a more compute efficient solution, we identify representative NL questions from PlotQA for training. We randomly sample 200K NL questions from the PlotQA training set and perform the ***generalize*** operation to replace schema specific information in each NL question with generalized tokens and to highlight its composition or structure. We replace the schema related entity (column headings) and values (column values) in the NL questions with more generic $< entity >$ and $< value >$ tags using sub-string matching. For example, the reasoning based question, depicted in Table 1 , is modified to: 'Is the difference between $< entity\_num >$ on $< value\_series >$ in $< value\_category >$ and $< value\_category >$ greater than the difference between $< entity\_num >$ on $< value\_series >$ in $< value\_category >$ and $< value\_category >$?', where 'the amount collected as tax' being a sub-string of the numerical column name of the schema, gets replaced with the generic token $< entity\_num >$ and the values of the category column, viz. '2003', '2007' and the series column 'goods' and 'exports' get replaced with $< value\_category >$ and $< value\_series >$, respectively. We further get the representations of these generalized NL questions using sentence-BERT (Reimers and Gurevych, 2019) and ***cluster*** them via DBSCAN [1] with *cosine similarity* as the similarity metric. As DBSCAN allows us to cluster data without specifying the number of clusters in advance, with $minpoints = 15$ and $\epsilon = 0.25$, we get 345 clusters for the 200K generalized NL questions. We then randomly ***sample*** 200 questions from each cluster to get 69K representative generalized NL questions. We fetch the corresponding original NL questions q_{rep} for these 69K questions along with their corresponding schema c_{rep} and answers a_{rep} to form a dataset of representative queries D_{rep}. A similar sampling strategy is applied on the validation split. Table 2 illustrates the statistics of the representative dataset.

4.2.4 FIBT Forward Pass

We train the forward model $M_{NL->P}$ with the training data D_{tr} by feeding the flattened schema, the table contents, the NL query with a separator token to the encoder and generate the SQL tokens at the output of the decoder in an auto-regressive fashion. The model is trained using cross entropy loss. We feed the NL queries q_{rep} from D_{rep} to $M_{NL->P}$ to generate the corresponding SQL programs (p_{rep}). We execute these SQL programs on the corresponding extracted chart schema s_{rep}. The programs which do not execute to the ground truth denotations are filtered, and the training data D_{tr} is augmented by the remaining pairs.

4.2.5 FIBT Backward Pass

We train the backward model $M_{P->NL}$ with the training data D_{tr} by feeding the flattened schema and the contents followed by the SQL program with a separator token to the encoder and generating the NL tokens at the output of the decoder in an auto-regressive fashion. We use p_{tr} in the training set D_{tr}, along with the SQL programs p_{filter} filtered in the prior iteration to ***induce*** the inverse probabilities of PCFG as explained in section 4.2.2. Here, the filtered programs are the ones whose equivalent NL questions do not match with PlotQA representative questions (explained later in this section). We sample SQL programs (p_{synth}) from the PCFG to be executed on s_{chart} and feed these synthesized SQL programs to the backward model $M_{P->NL}$ to generate the corresponding NL questions q_{synth}. We (i) transform q_{synth} by using the ***generalize*** operation, explained in section 4.2.3, (ii) extract the representation for q_{synth} using sentence-BERT (Reimers and Gurevych, 2019) and (iii) compare them with the representations of generalized representative queries (q_{rep}) using *cosine similarity*. NL questions having their maximum similarity score (***max_sim***) below a *threshold* are filtered. The SQL programs corresponding to the filtered NL questions are added to p_{filter}, representing queries not matching the PlotQA questions. With this filtering, we still observe some synthetic questions with semantic noise, meaning the semantics of the NL questions q_{synth} and the corresponding SQL programs p_{synth} do not match. (Shen et al., 2019) uses phrases of NL questions to estimate the operator candidates in the corresponding programs and thus reduce the search space of the semantic parser. We use a similar technique of phrase-operator ***look-up*** to further filter the synthetic query pairs. Given a q_{synth}, p_{synth} pair the ***look-up*** operation returns 'False' if the pre-defined (set-of) phrase(s) in NL query (q_{synth})

[1] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

NL Phrase		SQL Operator
ratio	↔	/
difference	↔	-
greater than	↔	>
less than	↔	<
total OR sum	↔	+ OR SUM
maximum OR highest	↔	MAX
minimum OR lowest	↔	MIN
average	↔	AVG
how many	↔	COUNT
median	↔	MEDIAN

Table 3: Mapping of NL Phrases and SQL Operators

do(es) not match with the (set-of) operator(s) in the corresponding SQL programs (p_{synth}) and returns 'True' otherwise. This matching is done following the pre-defined look-up dictionary with the phrase-operator mappings between the NL questions and the SQL programs (Table 3). This filtering helps to remove the semantically incorrect NL questions (q_{synth}), which have been generated by the backward model $M_{P->NL}$ for the synthetic SQL programs (p_{synth}). With this two level filtering, the training data D_{tr} is augmented with the remaining synthetic tuples <$s_{chart}, q_{synth}, p_{synth}$> and is further used to train the models in the next iteration. These added synthetic queries are closer to the PlotQA queries and thus help in adapting the models to answer PlotQA questions. For every iteration, the above defined *threshold* for similarity based query filtering is automatically set to a value for which the KL-divergence between the operator distributions of (i) PlotQA questions q_{chart} and (ii) the synthetic questions getting augmented to the training set ($q_{synth} - q_{filter}$), after filtering with the phrase-operator mappings (Table 3) is minimum. This ensures that the augmented synthetic query pairs, after filtering with the *threshold*, are closer to the required PlotQA questions.

5 Results and Discussion

CodeT5 (Wang et al., 2021c) provides the best results on the Spider dataset [2] in terms of execution accuracy. For chart QA, we are more interested in correctly computing the final answer (execution accuracy) than the intermediate logical form (exact match accuracy). Thus we choose CodeT5 based neural semantic parser as our forward model ($M_{NL->SQL}$) and CodeT5 based code summarization model [3] as the backward model ($M_{SQL->NL}$). The number of trainable parameters in the CodeT5 base model are 220M. We fine-tune the models with a batch size of 48 and a learning rate of 0.0001 and gradient accumulation step of 4 using the Adam optimizer on an NVIDIA Tesla V100 32GB GPU. The average run-time for training the model differs in each iteration as the number of training samples increase with the data we augment in each iteration. However, for inference on the PlotQA V2 Test Split, with the given hyper-parameter configuration, it takes ∼360 hours. For sampling the representative questions and SQL queries from the dataset and PCFG respectively, we use the random seed of 7. The training details of the chart extraction modules are provided in Appendix D. We use test accuracy as the evaluation metric, where for numeric answers with floating point values, we consider an answer to be correct if it is within the 5% range of the correct answer as followed by (Methani et al., 2020).

ANS Type	Test Queries	Human	Plot QA	T5	Ours ES	Ours OS
YN	72,968	**76.51**	**62.75**	62.38	43.21	44.13
FV	566,655	**59.97**	7.95	2.41	**63.91**	67.70
OV	2,919,769	**58.01**	14.95	0.003	**60.42**	85.35
Total	3,559,392	**58.70**	14.82	1.17	**60.44**	84.49

Table 4: Results on PlotQA V2 Reasoning Queries (% Test Accuracy), ANS: Answer, YN: Yes/No, FV: Fixed Vocabulary, OV: Open Vocabulary, ES: Extracted Schema, OS: Oracle Schema

Table 4 illustrates the results of the Q&A task over reasoning based queries in the PlotQA V2 test set (∼3.56M NL queries on ∼33.6K charts). Following the benchmark approaches (Methani et al., 2020; Masry et al., 2022), we report results on PlotQA V2 test set and not the validation set. As mentioned in (Methani et al., 2020) most human errors are due to numerical precision as it is difficult to visually identify the exact value from the chart even within a 5% margin. Our weakly supervised approach surpasses the baselines (PlotQA (Methani et al., 2020) and T5 (Masry et al., 2022)) by a large margin even exceeding the human baseline. We observe improvement in test accuracy results from 39.69% to 57.45% to 60.44% in the 1^{st}, 2^{nd} and 3^{rd} iterations of FIBT, respectively. This demonstrates the utility of the FIBT approach with the filtering and augmentation mechanisms used for capturing the relevant query compositions.

As per the definition provided by the authors of PlotQA, the fixed vocabulary comprises of the set of top 1000 frequently occurring answer words. Our approach yields superior performance for fixed

[2] Spider Leaderboard Dated: August 2022 https://yale-lily.github.io/spider

[3] https://huggingface.co/Salesforce/codet5-base-multi-sum

vocabulary (FV) and open vocabulary (OV) answers. Both FV and OV answers are numerical. Prior approach of end-to-end model predicting the final answer directly (T5 (Masry et al., 2022)) and approach which address queries with distinct answer types distinctly (PlotQA(Methani et al., 2020)), learn to distinguish the queries having YES/NO (binary) type of answers from other queries. Once this is learnt, any random guess of YES/NO as an answer to these queries would lead to a performance of 50%. On the other hand, our approach trains a single model to generate SQL programs for all queries with distinct answer types. For some cases the generated SQL program for questions with binary answers does not yield a binary result. Thus, for our approach, the random accuracy for Yes/No queries is not 50%. Moreover, the SQL programs for questions with YES/NO answers are more complex as compared to the SQL programs which leads to numerical answers (FV and OV questions) in terms of entailed compositions of primitive operations involving nesting, leading to harder synthesis. Also, our approach does not yield good performance for some of these questions with binary answers as there is no explicit mapping between the phrases in the NL query and the primitive operations involved in the SQL program. For example, for the questions: 'Do a majority of the years between 2013 and 2010 (inclusive) have a number of secure internet servers greater than 1.16? or 'Do the payments made towards primary income monotonically increase over the years?' or 'Is the payments made towards goods and services strictly less than the payments made towards primary income over the years ?' , the model finds it harder to learn to map the abstract phrases 'majority of' or 'monotonically increasing' or 'strictly less than' to a composition of primitive operations in the corresponding SQL programs as this knowledge is not explicitly provided. These are the reasons that the performance of our approach for queries with YES/NO answer type is inferior as compared to the other reasoning queries.

ChartQA (Masry et al., 2022) provides results on the complete PlotQA V2 test split (∼4.34 M questions) for all question types including structural, data retrieval and reasoning. The test accuracy of their best performing model (T5), trained on the complete PlotQA train set (∼20.25 M questions), end-to-end is 56.22%. For fair comparison with ChartQA (Masry et al., 2022), we train the T5 model in an end-to-end fashion (direct answer based supervision), with 69K representative questions (0.04%) of the PlotQA training set following the same input format as in ChartQA. We test the model on reasoning based questions in PlotQA V2 test data (∼3.56 M) to obtain the results depicted in Table 4. The T5 model can address the yes/no type of binary answers but struggles on questions with numerical answers (FV and OV). Effectively, as discussed earlier, once T5 has learnt to identify questions having binary answers, a random guess would lead to 50% accuracy. For T5, low performance on non-binary reasoning questions is expected because the end-to-end training in general struggles to perform complex reasoning in the latent space, and this is compounded by the very small amount of training data used for fair comparison. On the other hand, better performance of FV and OV answer type questions, underscores the efficacy of our approach to better handle complex numerical reasoning questions. Moreover, as we generate SQLs for NL, our approach is more interpretable, allowing users to understand the reasoning steps to get the final answer.

Apart from PlotQA (Methani et al., 2020), CharQA (Masry et al., 2022) is the only other dataset available. After thorough analysis of this dataset, we observed that the dataset contains samples with incorrect ground truth labels, spurious questions and Gold data tables with incorrect information. The details of our analysis is provided in Appendix G. Hence, we have not used CharQA (Masry et al., 2022) for benchmarking our approach.

To understand the impact of errors from vision based chart schema extraction on the downstream reasoning task, we perform an ablation to calculate the test accuracy of the reasoning task using the schema constructed with oracle extractions provided by PlotQA. We observe a 24.05% lower test accuracy with the extracted schema as compared to the oracle. We further analyze the results based on operators involved in the query. We observe that our approach works well not only for reasoning questions involving one primitive operators, but also for more complex questions involving composition of numerical operators such as (i) 'COUNT VALUES GREATER THAN AVERAGE' (For example, *For how many years, is the payments made towards primary income greater than the average payments made towards primary income over all*

years ?) or (ii) 'SUM GREATER THAN MAX' (For example, *Is the sum of the payments made towards goods and services in 2008 and 2010 greater than the maximum payments made towards primary income across all years ?*) or (iii) DIFFERENCE GREATER THAN DIFFERENCE' (For example, reasoning type of query mentioned in Table 1 for dot charts).

We further observe that our approach does not yield good results for NL questions involving nesting in the SQL program. For example, (i) queries computing a 'DIFFERENCE' between the 'MAXIMUM' and the 'SECOND MAXIMUM' values of the numerical column (For example, the reasoning type of query mentioned in Table 1 for (b) Horizontal bars). In the corresponding SQL program for such questions, to compute the 'SECOND MAXIMUM' value requires nesting, (ii) the queries which try to find if the 'SUM' of two numerical values is 'GREATER THAN' the 'SUM' of other two numerical values, for 'EVERY' value of a non-numerical column. These NL questions demand nested SQL programs which ensure the 'SUM GREATER THAN SUM' criteria is true for 'ALL DISTINCT' values of the non-numerical column. Table 7 shows that the defined PCFG allows synthesis of nested SQL programs. We observe that as such nested SQL programs are not present in the initial training set, the strategy of inducing inverse probabilities for the PCFG facilitates synthesis of nested SQL programs in the later iterations of FIBT. However, for most of such nested SQL programs the backward model fails to generate semantically meaningful NL questionsSuch noisy synthetically generated NL questions are filtered by our filtration strategy in the backward pass leading to fewer nested query samples in the training data which in turn cause low test accuracy for such questions.

6 Conclusion

We present an approach for QA on charts which addresses complex reasoning based questions that require a combination of natural language understanding, fine-grained perception, and analytical reasoning. We employ a pretrained semantic parser and FIBT we generate SQL programs for the NL questions without any program annotations. Our novel PCFG based approach helps the model to adapt to the given dataset's query compositions and domains, unseen in the bootstrapping data. As the future work we plan to extend our approach further to handle complex questions requiring nesting by using a hierarchical or grammar based program search technique.

7 Limitations

The focus of this paper is on complex reasoning type of questions. Our approach is not designed for structural (Table 1) or visual types of questions, pertaining to attributes of the visual elements of the plot such as color, size, spatial location. These questions are not useful for real-life applications, which require analysis on chart data to draw meaningful conclusions. Our existing approach is not designed to address such questions as the extracted schema only captures the underlying data of the chart, and not the visual entities present therein.

To have a good kick-start for the FIBT pipeline, we assume that the bootstrap SPIDER data covers the primitive SQL clauses, operators and functions required for the questions in the target dataset and there is some minimal overlap between the compositions of the SQL queries in the bootstrap SPIDER data and SQL queries required for the natural language questions in the target dataset.

In the current phrase-operator based filtering strategy only limited phrases are manually designed for the mapping of NL phrases and SQL operators (Table 3). We plan to make this approach more robust by using paraphrases or semantically similar phrases to manually designed phrases for mapping.

8 Ethical Considerations

PlotQA charts contain only factual information which is openly available in public domain and is not (i) specific to any individual or (ii) offensive. The solution provided in the paper is agnostic to the domain of the data. Like any QA task, to avoid the risks involved in critical domains such as finance, healthcare or medicine, we would have to calibrate the model or need human intervention, such that the errors are not propagated to the downstream tasks.

References

Rishabh Agarwal, Chen Liang, Dale Schuurmans, and Mohammad Norouzi. 2019. Learning to generalize from sparse and underspecified rewards. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 130–140. PMLR.

Priyanka Agrawal, Ayushi Dalmia, Parag Jain, Abhishek Bansal, Ashish Mittal, and Karthik Sankaranarayanan. 2019. Unified semantic parsing with weak supervision. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4801–4810, Florence, Italy. Association for Computational Linguistics.

Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. 2019. Character region awareness for text detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9365–9374.

Ritwick Chaudhry, Sumit Shekhar, Utkarsh Gupta, Pranav Maneriker, Prann Bansal, and Ajay Joshi. 2020. Leaf-qa: Locate, encode & attend for figure question answering. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3512–3521.

Jaemin Cho, Jie Lei, Hao Tan, and Mohit Bansal. 2021. Unifying vision-and-language tasks via text generation. In *International Conference on Machine Learning*, pages 1931–1942. PMLR.

DongHyun Choi, Myeong Cheol Shin, EungGyun Kim, and Dong Ryeol Shin. 2021. Ryansql: Recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases. *Computational Linguistics*, 47(2):309–332.

Pritha Ganguly, Nitesh Methani, Mitesh M Khapra, and Pratyush Kumar. 2020. A systematic evaluation of object detection networks for scientific plots. *arXiv preprint arXiv:2007.02240*.

Michael Glass, Mustafa Canim, Alfio Gliozzo, Saneem Chemmengath, Vishwajeet Kumar, Rishav Chakravarti, Avi Sil, Feifei Pan, Samarth Bharadwaj, and Nicolas Rodolfo Fauceglia. 2021. Capturing row and column semantics in transformer based question answering over tables. *arXiv preprint arXiv:2104.08303*.

Jiaqi Guo, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2021. Weakly supervised semantic parsing by learning from mistakes. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2603–2617.

Tong Guo and Huilin Gao. 2019. Using database rule for weak supervised text-to-sql generation.

Yinuo Guo, Hualei Zhu, Zeqi Lin, Bei Chen, Jian-Guang Lou, and Dongmei Zhang. 2020. Revisiting iterative back-translation from the perspective of compositional generalization. *arXiv preprint arXiv:2012.04276*.

Kaylin Hagopian, Qing Wang, Tengfei Ma, Yupeng Gao, and Lingfei Wu. 2019. Learning logical representations from natural languages with weak supervision and back-translation. In *Knowledge Representation & Reasoning Meets Machine Learning Workshop (KR2ML)*.

Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. *arXiv preprint arXiv:2004.02349*.

Vu Cong Duy Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. 2018. Iterative back-translation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 18–24, Melbourne, Australia. Association for Computational Linguistics.

Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. 2016. Dynamic filter networks. *Advances in neural information processing systems*, 29:667–675.

Kushal Kafle, Brian Price, Scott Cohen, and Christopher Kanan. 2018. Dvqa: Understanding data visualizations via question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656.

Kushal Kafle, Robik Shrestha, Scott Cohen, Brian Price, and Christopher Kanan. 2020. Answering questions about data visualizations using efficient bimodal fusion. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1498–1507.

Samira Ebrahimi Kahou, Vincent Michalski, Adam Atkinson, Ákos Kádár, Adam Trischler, and Yoshua Bengio. 2017. Figureqa: An annotated figure dataset for visual reasoning. *arXiv preprint arXiv:1710.07300*.

Dae Hyun Kim, Enamul Hoque, and Maneesh Agrawala. 2020. Answering questions about charts and generating visual explanations. In *Proceedings of the 2020 CHI conference on human factors in computing systems*, pages 1–13.

Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc Le, and Ni Lao. 2018. Memory augmented policy optimization for program synthesis and semantic parsing. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 10015–10027, Red Hook, NY, USA. Curran Associates Inc.

Qian Liu, Bei Chen, Jiaqi Guo, Zeqi Lin, and Jian-Guang Lou. 2021. Tapex: Table pre-training via learning a neural sql executor. *ArXiv*, abs/2107.07653.

Ahmed Masry and Enamul Hoque. 2021. Integrating image data extraction and table parsing methods for chart question answering. In *Chart Question Answering Workshop, in conjunction with the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–5.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. *arXiv preprint arXiv:2203.10244*.

Nitesh Methani, Pritha Ganguly, Mitesh M Khapra, and Pratyush Kumar. 2020. Plotqa: Reasoning over scientific plots. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1527–1536.

Sewon Min, Danqi Chen, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019. A discrete hard EM approach for weakly supervised question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2851–2864, Hong Kong, China. Association for Computational Linguistics.

Shubham Paliwal, Arushi Jain, Monika Sharma, and Lovekesh Vig. 2021. Digitize-pid: Automatic digitization of piping and instrumentation diagrams. *CoRR*, abs/2109.03794.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. *arXiv preprint arXiv:1508.00305*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *ArXiv*, abs/1908.10084.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

Tao Shen, Xiubo Geng, Tao Qin, Guodong Long, Jing Jiang, and Daxin Jiang. 2019. Effective search of logical forms for weakly supervised knowledge-based question answering. *arXiv preprint arXiv:1909.02762*.

Noah Siegel, Zachary Horvitz, Roie Levin, Santosh Divvala, and Ali Farhadi. 2016. Figureseer: Parsing result-figures in research papers. In *European Conference on Computer Vision*, pages 664–680. Springer.

Hrituraj Singh and Sumit Shekhar. 2020. Stl-cqa: Structure-based transformers with localization and encoding for chart question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3275–3284.

Ray Smith. 2007. An overview of the tesseract ocr engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, volume 2, pages 629–633. IEEE.

Bailin Wang, Mirella Lapata, and Ivan Titov. 2021a. Learning from executions for semantic parsing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2747–2759, Online. Association for Computational Linguistics.

Bailin Wang, Ivan Titov, and Mirella Lapata. 2019. Learning semantic parsers from denotations with latent structured alignments and abstract programs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3774–3785, Hong Kong, China. Association for Computational Linguistics.

Bailin Wang, Wenpeng Yin, Xi Victoria Lin, and Caiming Xiong. 2021b. Learning to synthesize data for semantic parsing. *arXiv preprint arXiv:2104.05827*.

Yue Wang, Weishi Wang, Shafiq Joty, and Steven CH Hoi. 2021c. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. *arXiv preprint arXiv:2109.00859*.

Tomer Wolfson, Daniel Deutch, and Jonathan Berant. 2021. Weakly supervised text-to-sql parsing through question decomposition.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. *arXiv preprint arXiv:2005.08314*.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. 2021. Grappa: Grammar-augmented pre-training for table semantic parsing. *ArXiv*, abs/2009.13845.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*.

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

Appendices

A Prior work on Table Q&A and Semantic Parsing

Current approaches for table QA use an end-to-end modeling approach for either: (i) directly generating the answer (TAPAS (Herzig et al., 2020)), (ii) generate a program which produces the answer

upon execution of the generated SQL (TABERT (Yin et al., 2020) and RCI (Glass et al., 2021)) or (iii) using a Language Model pre-training strategy (neural query execution engine in TAPEX (Liu et al., 2021) using synthetic SQL programs or semantic parser trained on synthetic NL-SQL pairs generated using a Synchronous Context Free Grammar (SCFG) in GraPPa (Yu et al., 2021). These approaches are mainly designed for handling data retrieval or simple aggregation type of queries defined in WikiSQL (Zhong et al., 2017) or Wiki Table (Pasupat and Liang, 2015) datasets. The complex reasoning type queries which are part of the PlotQA dataset involve nested arithmetic operations with self-joins as well as nesting in the conditional (WHERE) clauses. (Yin et al., 2020) applied their approach on the benchmark Spider Text-to-SQL dataset (Yu et al., 2018) but their results have been eclipsed by RYANSQL (Choi et al., 2021), whose decoder is specifically designed to address nested complex queries. GraPPa (Yu et al., 2021) achieves state of the art performance on the complex spider dataset after fine-tuning with spider program (SQL) annotations.

In our scenario there are no SQL program annotations for PlotQA queries, and only the output denotations are available. Since annotating a sufficiently large number of SQL programs is a resource heavy task, semantic parsers can be trained with execution output denotations under a weak-supervision setting. These execution denotations can be used to model a reward signal in order to train the underlying semantic parser (Zhong et al., 2017; Liang et al., 2018; Hagopian et al., 2019; Agrawal et al., 2019; Agarwal et al., 2019). They can also be used to train the semantic parser with a log Multiple log Marginal Likelihood (MML) objective by using a limited number of SQL programs as latent logical forms (Wang et al., 2019; Min et al., 2019; Wang et al., 2021a). While synthesizing SQL programs, the denotations can also be used to filter candidate programs with rule-based synthesis systems (Guo and Gao, 2019; Guo et al., 2021; Wolfson et al., 2021). The reward-based approaches face issues with a large program search space and possible spurious programs. Unlike the MML approaches, we do not use any gold annotated SQL programs from the dataset under consideration (PlotQA). Most of the above weak-supervision approaches do not explicitly handle modeling or synthesizing the specific complex program compositions pertinent to the dataset under consideration. On the other hand, our approach uses a publicly available semantic parsing dataset (Spider) as the bootstrapping data to initialize the parameters of the semantic parser, and then defines a novel PCFG-based strategy to adapt the models through FIBT to answer unseen complex reasoning queries in the PlotQA by effectively capturing the relevant query compositions.

B Text Role Region Extraction

The architecture consists a *Encoder-Decoder Module*, which the U-net (Ronneberger et al., 2015). The encoder filters out irrelevant information by squeezing the feature map to a latent space. The output of the decoder is appended along the channel dimension with the trigger patch of a text region detected by CRAFT, with highlighted patch contours and provided as an input to the *Trigger-Controller Module*, extracts features using a convolutional feature extractor followed by a Global Average Pooling (GAP) layer. The features of the trigger patch are concatenated with the extracted encoder output features and are fed to the controller module to generate dynamic kernels, which are used to generate the segmentation map (Jia et al., 2016). The dynamic kernel output is also given to a fully connected linear layer to determine the text role of the region. Thus, the trigger-controller module exploits the spatial relationships between text-roles to generate dynamic kernels and obtain text-role specific segmentation maps from the decoded image. The whole network is trained with cross entropy loss on text-role class labels and text-role segmentation map (all ground truth text-regions corresponding to that text-role). During inference, given a trigger patch for an image of a detected text-region belonging to an unknown text-role, the model provides the actual text-role classification output and the segmentation map of the text-region of that text-role. Trigger patches overlapping with detected text-role regions are removed before repeating the process for the remaining trigger patches. This process may lead to multiple segmentation maps for each text-role, over which a union operation is performed.

C Noise Correction

To handle false positive and false negative detection of Numerical-axis ticks, we find the mode (M) of differences between the (X/Y) coordinates for the consecutive ticks. We remove or add ticks where

IOU	@0.90										@0.75	@0.50
Existing Models (Ganguly et al., 2020)	**Bar**	**Dot-line**	**Leg Lbl**	**Leg PV**	**Title**	**X-Lbl**	**X-Ticks**	**Y- Lbl**	**Y- Ticks**	**mAP**	**mAP**	**mAP**
FrCNN (FPN+RA)	87.59	31.62	79.05	66.39	0.22	69.78	88.29	46.63	84.60	61.57	69.82	72.18
FrCNN (RA)	63.86	14.79	70.95	60.61	0.18	83.89	60.76	93.47	50.87	55.49	89.14	96.80
FrCNN (FPN+RA)	85.54	27.86	93.68	96.30	0.22	99.09	96.04	99.46	96.80	77.22	94.58	**97.76**
PlotNet	**92.80**	**70.11**	**98.47**	**96.33**	**99.52**	**97.31**	**94.29**	**97.66**	**94.48**	**93.44**	**97.93**	**98.32**
Ours (Train: All)	**89.67**	**69.13**	**99.89**	**98.67**	**99.99**	**99.90**	**99.45**	**99.89**	**97.69**	**94.92**	**95.80**	96.70
Ours (Train: 4K)	89.67	69.13	96.31	96.31	99.63	96.35	96.84	99.48	96.58	92.86	93.66	94.50

Table 5: Chart Extractions on the PlotQA dataset with mAP scores (in %). Leg: Legend, Lbl: Labels, PV: Preview

Method	PlotQA	Ours
Title	94.60	**99.69**
X-axis Label	95.50	**99.73**
Y-axis Label	97.07	**99.59**
Legend Labels	91.11	**98.13**
X-tick Labels	91.38	**97.62**
Y-tick Labels	88.07	**95.94**

Table 6: OCR Module Accuracy

the difference in the consecutive ticks is more or less than the mode, respectively. We add a dummy value 'x' for the newly added tick, if any, which is handled during correction. We replace a non-numerical tick-value detection, if any, by using M as an offset to the the neighboring tick value. To correct the tick-values not adhering to a progression followed by the majority values, we consider a tick-value as an 'anchor' (correct value) and calculate the other values adding and/or subtracting M from this anchor. We compute the 'gain' with respect to this anchor to be the intersection of the extracted values and calculated values. We repeat this process by considering distinct numerical-axis values as anchors. For the 'anchor' giving us the maximum 'gain', the corresponding calculated values are considered to be the correct set of numerical-axis tick values. Some charts (Figure 1 (a)) use scientific notation for denoting large numerical values which are converted to float values.

D Training Details for Chart Extraction Models

For the chart type classification ResNet-34 is fine-tuned for 2000 steps. We use the Adam optimizer, a learning rate of 0.0005, and a batch size of 8. The model yields 99.91% test accuracy. For text region and role detection, we use the pre-trained VGG19 and train with a batch size of 8, for 1 epoch, using the Adam optimizer with an initial learning rate of 0.0005. As the data is skewed for axes-labels, while creating training tuples, we under-sample for this text-role to avoid class imbalance.

E Data Extraction

We use interpolation and extrapolation to calculate a numerical value associated with every pixel on the numerical-axis by using numerical-axis tick coordinate pixels as reference. For every data point (pivot-point in case of dots and line charts, bar-tops in case of bars) detected we assign it: (i) a series by matching its style with the style of a legend label, (ii) a category by matching its coordinate with the category tick, and (iii) a value of the pixel on the numerical axis whose coordinate matches with it. Thus, we extract the data in the form of a set-of tuples <$category\_label$, $series\_label$, $numerical\_value$>. For the category for which no pivot point or bar is detected for a series due to VED errors, we consider its value to be zero. We finally define a tabular schema with the column names as category axis label (category column), series (series column), and a string formed by concatenating the chart title with the numerical axis label (numerical column). All the spaces in the column names are replaced by underscores to make them SQL compatible. We insert the extracted tuples as rows in the tabular schema. The generic schema obtained for the chart in Figure 1(b) is shown in Figure 2. Charts containing only one series have a schema with only category and numerical columns.

Improvements over FIBT Iterations The generated SQL programs for the NL queries requiring certain compositions of primitive operations got corrected after the iterations of back-translation. For example, the required composition of the NL query 'In how many years, is the amount spent in making social contributions greater than the average amount spent in making social contributions taken over all years ?' is 'Count Greater Than Average'. After the first iteration the generated SQL program for this query is "Select t1.social_contributions > t2.social_contributions from table_data t1, table_data t2 where t1.year = '2010' and t2.year = '2010' ", which got corrected to "select count(*) from table_data where social_contributions > (select avg (social_contributions) from table_data)", after the second iteration. This improvement is because of the data augmented by PCFG-based synthetic queries, which contained these SQL queries having compositions not present in the original bootstrapping data (details in Section 4.3). Some additional improvements over the iterations are because of the coverage of the additional (chart) domains, which

sql	→	sel_num_col | sel_col | sel_arth
sel_num_col	→	"SELECT" agg "(" "num_col_name" ")" from "WHERE" cond_series | cond_cat
sel_col	→	sel_cat_col | sel_series_col "WHERE" cond_num ("AND" cond_series | cond_cat $)^{0-1}$
sel_col	→	sel_cat_col | sel_series_col "ORDER" "BY" "num_col_name" "DESC" | "ASC" "LIMIT" "1"
sel_cat_col	→	"SELECT" "DISTINCT" ("COUNT" $)^{0-1}$ "(" "cat_col_name" ")" from
sel_series_col	→	"SELECT" "DISTINCT"("COUNT" $)^{0-1}$ "(" "series_col_name" ")" from
sel_arth	→	"SELECT" agg "(" "num_col_name" ")" arth agg "(" "num_col_name" ")" from ("WHERE" cond_series | cond_cat $)^{0-1}$
sel_arth	→	"SELECT" "DISTINCT" "(" "t1." "num_col_name" arth "t2." "num_col_name" ")" from_2 "WHERE" "t1." cond_cat | cond_series "AND" "t2." cond_cat | cond_series ("AND" "t1." cond_series | cond_cat "AND" "t2." cond_series | cond_cat)$^{0-1}$
sel_arth	→	"SELECT" "DISTINCT" "(" "t1." "num_col_name" arth "t2." "num_col_name" ")" arth agg "(" "t3." "num_col_name" ")" from_2 "," "table_data" "t3" "WHERE" "t1." cond_cat "AND" "t2." cond_cat ("AND" "t1." cond_series "AND" "t2." cond_series "AND" "t3." cond_series)$^{0-1}$
sel_arth	→	"SELECT" "DISTINCT" "(" "t1." "num_col_name" arth "t2." "num_col_name" ")" arth "(" "t3." "num_col_name" arth "t4." "num_col_name" ")" from_2 from_4 "WHERE"" t1." cond_series | cond_cat "AND" "t2." cond_series | cond_cat "AND" "t3." cond_series | cond_cat "AND" "t4." cond_series | cond_cat
from	→	"FROM" "table_data"
from_2	→	"FROM" "table_data" "AS" "t1" "," "table_data" "AS" "t2"
from_2	→	"FROM" "table_data" "AS" "t1" "JOIN" "table_data" "AS" "t2" "ON" "t1." "cat_col_name" | "series" "=" "t2." "cat_col_name" | "series"
from_4	→	"," "table_data" "t3" "," "table_data" "t4"
cond_num	→	"num_col_name" op "num_val" | op "(" sel_num_col ")" | "NOT" "IN" "(" sel_num_col ")"
cond_series	→	"series" "=" "'""series_col_val""'" | "cat_col_name" "IN" "(" sel_cat_col ")"
cond_cat	→	"cat_col_name" "=" "'""cat_col_val""'" | "series" "IN" "(" sel_series_col ")"
agg	→	"SUM" | "MIN" | "MAX" | "AVG" | "MEDIAN"
arth	→	"<" | ">" | "/" | "-" | "+"
op	→	"=" | "<" | ">"

Table 7: Probabilistic Context Free Grammar (PCFG)

are not present in the initial bootstrapping data (having queries covering the SPIDER database domains), but got covered in the PCFG based synthetically generated queries on the PlotQA chart schema, addressing the domain shift.

F Results of Plot Extraction

For plot extractions we prefer mAP @0.90 IOU over mAP @0.75 and mAP@0.50 IOU as the evaluation metric as we require precise fine-grained extractions else the resulting data errors will propagate to the downstream QA task. For OCR we use accuracy (1 - Word Error Rate (WER)) as the evaluation metric. Table 5 illustrates the SOA results on plot extraction by our approach. We have state-of-the-art results with for Chart Extraction yielding 94.92% mAP @0.90 IOU when trained with all PlotQA (157K) images, beating the baseline (Ganguly et al., 2020) by 1.48%. mAP @0.90 IOU. The extraction of dot/line regions is challenging because of their small size, sparse distribution and eclipsed or intersected dots/lines of distinct series. Table 6 depicts the SOA results of the OCR module. The PlotQA Oracle refers to the results with the OCR model applied to the ground truth text-regions. Our predicted text-detections followed by OCR outperform both the baselines from PlotQA (Methani et al., 2020), yielding State-of-the-Art results.

G ChartQA Dataset Analysis

After thorough analysis of CharQA (Masry et al., 2022) dataset, because of the following observations we have not used it for benchmarking: (i) Samples with incorrect Ground Truth (GT) labels: Few examples: (a) for the question '*In what year did Nicaragua have the highest risk score of money laundering and terrorist financing?*' on the chart '*two_col_102453.png*', the actual answer is '2020', and the provided GT is '2015', (b) for the question *'What's the ratio of the lowest value of green bars and blue bars?'* on the chart *'1392.png'*, the

actual answer is '2.41' and the GT is '1.216', (c) For the question 'In republicans what is the difference between the more likely and less likely?' on the chart '9987.png' the ground truth label is '21', where as based on the question the ground truth label should be '-21'. (ii) Samples with spurious questions: The contents of the question are not relevant to the data present in the chart. Few examples: (a) For the question *'What was the third most popular brand on Foursquare?'* on the chart *'two_col_80744.png'*, 'Foursquare' is itself an individual brand depicted in the chart and no other information about Foursquare is provided. and the GT is 'MTV' which in itself is a separate brand and not related to Foursquare. (b) the question *'How many people checked in to New Delhi on Facebook between June and August 2017?'* posed on chart *'two_col_5556.png'* has some terms such as Facebook or specified Month/Year which are not present in the chart, (iii) Gold data tables[4] with incorrect information: Gold data tables crawled through web source *'PEW'*[5] have incorrect information, which leads to predicted answers. In the test set 310 (20.54%) samples have all the data values to be zeros and others (*'4931.png'*, *'2721.png'*, *'11627839005738.png'*) have floating point errors. For example, the Gold data tables for charts '4931.png', '2721.png', '11627839005738.png' are spurious. Due to this incorrect Gold Table contents even though the GT labels corresponding to the question to these charts is correct, with incorrect data points, the resulting answer does not match the label. Due to the above listed observations, we have not used this dataset for the benchmarking purpose.

[4] https://drive.google.com/file/d/17-aqtiq_KJ16PIGOp30W0y6OJNax6SVT/view

[5] https://github.com/vis-nlp/ChartQA/issues/8