FigureSeer: Parsing Result-Figures in Research Papers

Noah Siegel[1,2](✉), Zachary Horvitz[1,2], Roie Levin[1,2], Santosh Divvala[1,2],
and Ali Farhadi[1,2]

[1] Allen Institute for Artificial Intelligence, Seattle, USA
noahs@allenai.org
[2] University of Washington, Seattle, USA
http://allenai.org/plato/figureseer

Abstract. 'Which are the pedestrian detectors that yield a precision above 95 % at 25 % recall?' Answering such a complex query involves identifying and analyzing the results reported in figures within several research papers. Despite the availability of excellent academic search engines, retrieving such information poses a cumbersome challenge today as these systems have primarily focused on understanding the text content of scholarly documents. In this paper, we introduce FigureSeer, an end-to-end framework for parsing result-figures, that enables powerful search and retrieval of results in research papers. Our proposed approach automatically localizes figures from research papers, classifies them, and analyses the content of the result-figures. The key challenge in analyzing the figure content is the extraction of the plotted data and its association with the legend entries. We address this challenge by formulating a novel graph-based reasoning approach using a CNN-based similarity metric. We present a thorough evaluation on a real-word annotated dataset to demonstrate the efficacy of our approach.

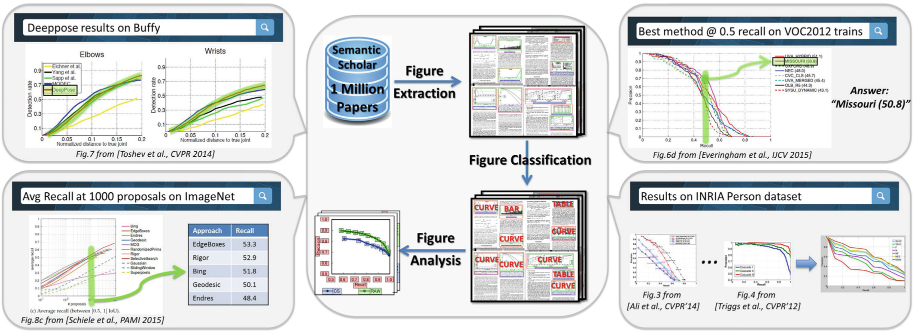
1 Computer Vision for Scholarly Big Data

Academic research is flourishing at an unprecedented pace. There are already over 100 million papers on the web [1] and many thousands more are being added every month [2]. It is a Sisyphean ordeal for any single human to cut through this information overload and be abreast of the details of all the important results across all relevant datasets within any given area of research. While academic-search engines like Google Scholar, CiteSeer, etc., are helping us discover relevant information with more ease, these systems are inherently limited by the fact that their data mining and indexing is restricted to the text content of the papers.

Research papers often use figures for reporting quantitative results and analysis, as figures provide an easy means for communicating the key experimental observations [3]. In many cases, the crucial inferences from the figures are often

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-46478-7_41) contains supplementary material, which is available to authorized users.

B. Leibe et al. (Eds.): ECCV 2016, Part VII, LNCS 9911, pp. 664–680, 2016.
DOI: 10.1007/978-3-319-46478-7_41



Approach	Recall
EdgeBoxes	53.3
Rigor	52.9
Bing	51.8
Geodesic	50.1
Endres	48.4

Fig. 1. FigureSeer is an end-to-end framework for parsing result-figures in research papers. It automatically localizes figures, classifies them, and analyses their content (center). FigureSeer enables detailed indexing, retrieval, and redesign of result-figures, such as highlighting specific results (top-left), reformatting results (bottom-left), complex query answering (top-right), and results summarization (bottom-right).

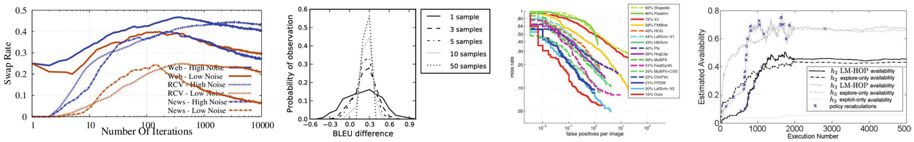
not explicitly stated in text (as humans can easily deduce them visually) [4]. Therefore failing to parse the figure content poses a fundamental limitation towards discovering important citations and references. This paper presents FigureSeer, a fully-automated framework for unveiling the untapped wealth of figure content in scholarly articles (see Fig. 1).

Why is figure parsing hard? Given the impressive advances in the analysis of natural scene images witnessed over the past years, one may speculate that parsing scholarly figures is a trivial endeavor. While it is true that scholarly figures are more structured than images of our natural world, inspecting the actual figure data exposes a plethora of complex vision challenges:

Strict requirements: Scholarly figures expect exceptional high-levels of parsing accuracy unlike typical natural image parsing tasks. For example, in Fig. 2(c), even a small error in parsing the figure plot data changes the ordering of the results, thereby leading to incorrect inferences.

High variation: The structure and formatting of scholarly figures varies greatly across different papers. Despite much research in engendering common design principles, there does not seem to be a consensus reached yet [5,6]. Therefore different design conventions are employed by authors in generating the figures, thereby resulting in wide variations (see Fig. 2).

Heavy clutter and deformation: Even in the best case scenario, where figures with a common design convention are presented, there still remains the difficulty of identifying and extracting the plot data amidst heavy clutter, deformation and occlusion within the plot area. For example, in Fig. 2(d), given just the legend symbol template for 'h_3 LM-HOP availability' method, extracting its plot data is non-trivial due to the heavy clutter and deformation (also see Fig. 4).



(a) Fig.6 from [Joachims et al., ICML 2013] *(b) Fig.7 from [Smith et al., ACL 2013]* *(c) Fig.7 from [Zitnick et al., CVPR 2013]* *(d) Fig.2 from [Veloso et al., AAAI 2011]*

Fig. 2. There is high variation in the formatting of figures: some figures position the legend within the plot area, while others place it outside. Within the legend, some figures have symbols on the right of the text, while others on the left. The presence of heavy occlusions and deformations also poses a challenge.

While color is an extremely valuable cue for discriminating the plot data, it may not always be available as many figures often reuse similar colors (see Fig. 2), and many older papers (even some new ones [7,8]) are published in grayscale. Moreover, unlike natural image recognition tasks where desired amount of labeled training data can be obtained to train models per category, figure parsing has the additional challenge where only a single exemplar (i.e., the legend symbol) is available for model learning. All these challenges have discouraged contemporary document retrieval systems from harvesting figure content other than simple meta-data like caption text.

Overview: The primary focus of our work is to parse result-figures within research papers to help improve search and retrieval of relevant information in the academic domain. The input to our parsing system is a research paper in *.pdf* format and the output is a structured representation of all the results-figures within it. The representation includes a detailed parse of each figure in terms of its axes, legends, and their corresponding individual plot data. We focus our attention on result-figures as they summarize the key experimental analysis within a paper. More specifically, within our corpus of papers, we found 2D-graphical plots plotting continuous data (such as precision-recall, ROC curves, etc.) to be most popular and frequent.

In this paper, we present a novel end-to-end framework that automatically localizes all figures from a research paper, classifies them, and extracts the content of the result-figures. Our proposed approach can localize a variety of figures including those containing multiple sub-figures, and also classify them with great success by leveraging deep neural nets. To address the challenges in parsing the figure content, we present a novel graph-based reasoning approach using convolutional neural network (CNN) based similarity functions. Our approach is attractive as it not only handles the problems with clutter and deformations, but is also robust to the variations in the figure design. As part of this work, we also introduce thorough evaluation metrics, along with a fully-annotated real-world dataset to demonstrate the efficacy of our parsing approach. Finally, to demonstrate the potential unleashed by our approach, we present a query-answering system that allows users to query figures and retrieve important information.

In summary, our key contributions are: (i) We introduce and study the problem of scholarly figure parsing. (ii) We present a novel end-to-end framework that automatically localizes figures, classifies them, and analyzes their content. (iii) We present a thorough evaluation on a real-word dataset to demonstrate the efficacy of our approach. (iv) We demonstrate the utility of our parsing approach by presenting a query-answering system that enables powerful search and retrieval of results in research papers using rich semantic queries. (v) Finally, we release a fully-annotated dataset, along with a real-world end-to-end system for spurring further research. We hope our work will help kick-start the challenging domain of vision for scholarly big data.

2 Related Work

Figure Extraction and Classification Localizing objects within natural images is a well-studied problem in computer vision. However, localizing figures within research papers has only recently become an area of interest. While many 'off-the-shelf' tools exist that can extract embedded images from *.pdf* files [9], these tools neither extract *vector-graphic* based images nor the associated captions of the figures. Recent works [10–12] have explored figure extraction by processing the PDF primitives. The work of [11] is interesting as it extracts a wide variety of figures along with their captions. In this paper, we build upon this work by augmenting their method with sub-figure localization.

Classifying scholarly figures has also recently become an area of research interest [6,13]. The work of [6] used a visual bag-of-words representaiton with an SVM classifier for classifying figures. In this paper, we leverage the recent success of CNNs and present an improved classifier that surpasses the state-of-the-art performance.

Figure Analysis. Much attention in the document analysis community has been devoted towards analyzing the document text content [14–17], but analyzing the figure content within the documents has received relatively little focus. Given the challenges in figure parsing (see Sect. 1), most works have either resorted to manual methods [18,19] or restricted their focus to limited domains with strong assumptions [6,13,20].

In [20], graphical plots were assumed to plot only a single variable. Further, rather than extracting the plot data, their focus was limited to recognizing the intended message (e.g., rising trend, falling trend, etc.,) of the plot. [6] presented a simple method for parsing bar charts. Their method located the bars by extracting connected components and then used heuristics to associate the marks with the axes. While their method achieved impressive results, its focus was limited to bar plots with color and those having a linear-axis scale. Further, their method failed to detect and leverage the legend information. Our proposed method circumvents these limitations, and thereby helps improve the generalizability and robustness of their bar parser as well.

Query-Answering. Challenges with figure parsing have discouraged contemporary document retrieval systems from harvesting the figure content. Most existing academic search engines respond to queries by only using the textual meta-data content about the figures, such as the caption text, or their mentions in the body text [17,21–23]. While there exists a few methods that have considered using content from tables [15], to the best of our knowledge, there does not exist any method to query research papers by understanding figure content.

3 Figure Parsing Approach

Our figure parser first extracts figures from a given *.pdf* file (Sect. 3.1), then segregates the figures (Sect. 3.2), and finally analyzes the content of the result-figures (Sect. 3.3). Figure 1(center) gives an overview of our overall framework.

3.1 Figure Extraction

Given the deluge of papers, it is desirable to have a scalable and robust approach for extracting figures. We leverage the work of [11] for figure extraction where a method for automatically localizing figures (using bounding boxes) along with their captions was presented. Their method analyzes the structure of individual pages by detecting chunks of body text, and then locates the figure areas by reasoning about the empty regions. The method was demonstrated to achieve high accuracy (F1 > 0.9), while being computationally efficient ($\sim$1 sec/paper).

A key limitation of [11] is its inability to localize individual figures within a figure containing multiple subfigures. Research papers often employ subfigures to report related sets of experimental results together. Given the frequent use of subfigures, we use an iterative method for separately localizing subfigures. More specifically, given an extracted figure, we iteratively decompose it into subfigures by identifying valid axis-aligned splits using the following criteria: (i) Both resulting regions must have an aspect ratio between $1 : c_1$ and $c_1 : 1$ ($c_1 = 5$); (ii) The ratio of the areas of the resulting regions must be between $1 : c_2$ and $c_2 : 1$ ($c_2 = 2.5$). The first criterion ensures that we avoid splits resulting in extremely narrow subfigures (that often happens by accidentally splitting off an axis or legend label). The second criterion enforces a weak symmetry constraint between the resulting halves (as subfigures are all often approximately of the same size). Our proposed method is simple, efficient and achieves promising results (see supplementary for more details).

3.2 Figure Classification

While graphical plots are the most common result-figures within research papers, there are often other figure types (natural images, flow charts, etc.,) found amongst the extracted figures. Therefore, we use a figure classifier to segregate the different figures and identify the relevant graphical plots. Convolutional Neural Networks (CNNs) have recently emerged as the state-of-the-art for classifying natural image content. Encouraged by the positive results in the domain of natural images, here we study their performance at large-scale figure classification.

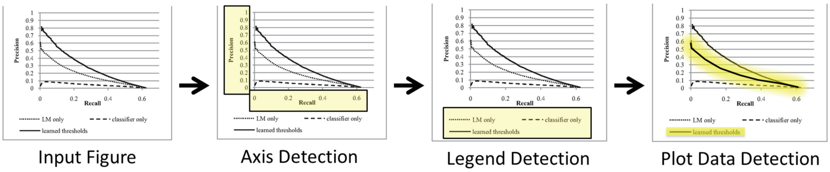


Fig. 3. Our figure analyzer first parses the figure axes, then the legend contents, and finally extracts and associates the plotted data to their legend entries.

We evaluate two network architectures: AlexNet [24] and ResNet-50 [25]. Both networks were pretrained on the 1.2 million images from ImageNet [26] and then fine-tuned for our figure classification task. It is well known that CNNs consume and benefit from large training sets. Sect. 4 describes the dataset collected for training our network.

3.3 Figure Analysis

Given all the segregated graph plots, we next analyze their content to obtain their corresponding detailed structured representation. This involves analyzing the figure axes, the figure legend, and the figure plot-data (see Fig. 3).

Parsing Axes. Parsing the axes involves determining their position, labels, and scales. Detecting the axes position helps in identifying the bounds of the plot area. Therefore we first detect the axes by finding all text boxes within the figure that correspond to the axis *tick* labels (e.g., ‘0’, ‘0.2’, ‘0.4’, ‘0.6’ on x-axis in Fig. 3). This is done by detecting series of (numeric) text boxes aligned in a straight line (representing the axis tick labels). More specifically, the y-axis (or x-axis) is determined by detecting the largest number of (numeric) text boxes that all share a common x (or y) pixel coordinate, breaking ties by choosing the leftmost qualifying x (or y) coordinate.

Each axis is almost always associated with a textual label that helps towards the interpretation of the graphical plot (e.g., the label ‘Precision’ for y-axis in Fig. 3). Given the common convention of placing the axis label in the immediate vicinity of the axis-tick labels, we detect the y-axis label by identifying the rightmost textbox to the left of the y-axis tick labels, and the x-axis label by finding the highest textbox below the x-axis tick labels.

While most plots use a linear axis scale, it is not uncommon for figures to have a logarithmic scale. Therefore we determine the axis scale (linear, logarithmic) by fitting separate regressors [27] (linear and exponential link functions) to model the data values, and then pick the model with the lowest deviance under a threshold. The regressors map the axis tick label values to their corresponding pixel coordinate values. These models are in turn used for transforming all the plotted data from their pixel-coordinate scale to their data-coordinate scale.

Parsing Legend. Graphical plots always use a legend as a guide to the symbols used when plotting multiple variables. Typically the legend has entries consisting of (*label*, *symbol*) pairs, where the labels are the variable names (e.g., ‘classifier only’ in Fig. 3) and the symbols give an example of its appearance. As highlighted in Sect. 1, there is huge variation in the placement and format of legends across figures. Legend entries may either be arranged vertically, horizontally, or in a rectangle, and they may be found either outside the plot area or anywhere inside (see illustration in supplementary). Further, the legend symbols may be placed either to the right or left of the legend labels, and may have varying lengths with spaces (e.g. the dashed symbol for ‘classifier only’ in Fig. 3). To address this challenge, our legend parser first identifies the legend labels, and then locates their corresponding symbols.

We pose the problem of legend label identification as a text classification problem, i.e., given a text box within the figure, is it a legend label or not? For classification, we use a random-forest classifier [28] with each textbox represented using a six-dimensional feature $f = \{t_x, t_y, t_l, t_n, t_{\#v}, t_{\#h}\}$, where t_x, t_y refer to the normalized x, y center coordinates of the text box, t_l is the text string length, t_n is a Boolean indicating the text string to be numeric or not, and $t_{\#v}, t_{\#h}$ denote the number of other vertically and horizontally aligned textboxes.

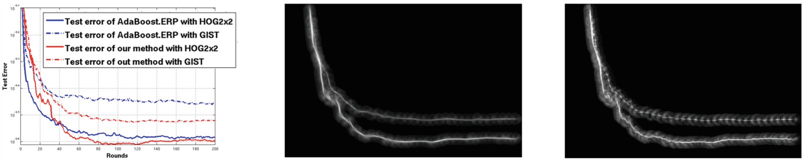
For localizing the symbols s corresponding to the identified legend labels t, we first need to determine their side (i.e., left or right of the text). This is done by generating two candidate rectangular boxes to the left and right of each label (s_{left}, s_{right}) with height $h = t_h$ (i.e., textbox height) and width $w = k * t_h$ $(k = 10)$. Each candidate is then assigned a score corresponding to its normalized non-background pixel density. The candidate scores across all labels on each side (i.e., left or right) are multiplied and the side with the highest score product is chosen. The selected candidate boxes are subsequently cropped to obtain the final symbol bounds (see supplementary for more details).

Parsing Plot-data. Our approach to parsing the plotted data is to formulate it as an optimal path-finding problem: given a legend symbol template s and the extent of the plot area $W_{n\times m}$, find its optimum path $P_s = \{\mathbf{x}_i\}_{i=1}^n = \{(x_i, y_i)\}_{i=1}^n$, such that the following energy function is optimized:

$$E(P_s) = \sum_{i=1}^{n} \phi_i(\mathbf{x}_i) \quad + \sum_{i=1,j=i+1}^{n-1} \phi_{ij}(\mathbf{x}_i, \mathbf{x}_j),$$
$$s.t., \ \forall i, \ 1 \leq y_i \leq m, \ 1 \leq x_i \leq n, \ x_{i+1} = x_i + 1.$$

The unary potential $\phi_i(\mathbf{x}_i) = \alpha f(\mathbf{x}_i)$ measures the likelihood of a pixel $\mathbf{x}_i$ to belong to the path given its features $f(\mathbf{x}_i)$. The pairwise potential $\phi_{ij}(\mathbf{x}_i, \mathbf{x}_j) = \beta f(\mathbf{x}_i, \mathbf{x}_j)$ is used to encourage smooth transitions between adjacent pixels $(\mathbf{x}_i, \mathbf{x}_j)$ by setting the pairwise features based on their slope i.e., $f(\mathbf{x}_i, \mathbf{x}_j) = (y_i - y_j)^2$. Inference under this model translates to finding the highest scoring path, which can be done using dynamic programming in linear time [29].

For learning the model weights (α, β), we use a rank SVM formulation [30]. The training examples for the ranker are pairs of the form (P_s, P'_s) with the goal of ranking all sub-optimal paths P_s to be lower than the ground-truth path P'_s. A path is defined to be suboptimal if its score (using our evaluation metric as



Input Figure
(Fig from [Koller et al., ICML'11])

Convolution map for
red solid-line symbol

Convolution map for
red dashed-line symbol

Fig. 4. Similarity maps using standard convolution for two different symbols. Simply convolving the symbol template with the plot area fails to discriminate well between the plots. For e.g., the red dashed-line symbol obtains a high response on patches corresponding to the red solid-line. Our approach circumvents this problem by learning similarity functions using CNNs.

defined in Sect. 5) is lower than a threshold. We use a bootstrapping procedure that mines hard negative examples to train the ranker [31].

Feature representation $f(\mathbf{x}_i)$ plays a crucial role towards the success of our model. Given the presence of heavy occlusions and deformations of the plotted data, simply convolving the legend symbol s with the plot area W using standard gradient-based features [31] fails to yield a robust representation (see Fig. 4). To address this challenge, we instead derive our feature representation by learning a feature function using CNNs [32,33] that allows us to implicitly model the various patch transformations.

We learn an embedding of image patches to a low dimensional feature vector using a Siamese network based on [32]. Each branch of the network consists of 3 convolutional layers followed by a fully connected layer, with ReLU and max pooling between layers. The input of each branch is a 64×64 grayscale image patch. The final layer of each branch projects this input to a 256 dimensional feature vector. Each training example consists of a legend symbol patch and a plot patch. The legend symbol patch is generated by padding and/or cropping the annotated legend symbol to 64×64 pixels. For positive examples, the plot patch is a 64×64 patch centered on a point on the symbol's corresponding ground-truth trajectory in the plot area. Negative pairs are obtained by sampling plot patches both randomly and from other symbol trajectories.

The network is trained using a contrastive loss function [34]. We augment our data by flipping both symbol and plot patches in pairs horizontally. During training, we use two feature networks with the constraint that the two networks share the same parameters. At testing, we use a single network where we independently pass a symbol patch s as well as patches from the plot area W through it and obtain their output representations. The final feature map for the symbol s is then estimated as the L_2 similarity between the output representations.

Along with these CNN-based similarity features, we also use the following pixel-based similarity features to define our unary features $f(\mathbf{x}_i)$: (i) *symbol convolution*: rotationally convolving the symbol patch s with the plot area W, which helps in capturing local visual similarities [35]; (ii) *connected-component* *size*: finding regions within the plot area W having similar connected-component statistics as the symbol patch s, which helps in differentiating patterns of dashes with varying lengths or thickness [36]; (iii) *color match*: finding regions in the plot area W that have the same color as the symbol patch s, which helps in differentiating unique colored plots; (iv) *breathing*: a constant valued feature map, which helps in handling plots whose domain does not cover the full extent of the x-axis (see supplementary materials for more details).

Implementation details: Training our similarity network takes 20 h on a Titan X GPU using Caffe [37]. Parsing a new figure takes 8 sec on an Intel Xeon E5-1630 CPU, and 40 sec for generating the CNN feature on our GPU.

4 FigureSeer Dataset

The availability of a standardized dataset plays a crucial role in training and evaluating algorithms as well as in driving research in new and more challenging directions. Towards this end, we have built an annotated figure parsing dataset using over 20,000 papers covering five different research areas (CVPR, ICML, ACL, CHI, AAAI) obtained from the 1 million CiteSeerX papers [17] indexed by Semantic Scholar [38]. Processing the 20,000 papers using the method of [11] yielded over 60000 figures. All these figures were then annotated using mechanical turk [39] for their class labels (scatterplot, flowchart, etc.,).

Of all the figures annotated as graph plots, we randomly sampled over 600 figures for further detailed annotations. Labelling the figures with their detailed annotations, i.e., axes, legends, plot data, etc., is a complex and multi-step task, making it more difficult to crowdsource over mechanical turk [40]. Therefore we trained in-house annotators to label the figures using a custom-made annotation tool. For each figure, the annotators annotated the axes (position, title, scale), the legend (labels, symbols), and the plotted data for each legend entry. Annotating the figures yielded 1272 axes, 2183 legend entries and plots. 55 % of the figures are colored, while 45 % are grayscale. An overview video of our annotation interface as well as our complete annotated dataset is available on our project page.

5 Figure Parsing Results

Figure Classification. We used the 60000 figures from our dataset to study the performance of our network. The figures were randomly split into two equal halves (30000 each) for training and testing. Table 1 summarizes our results in comparison to the previous state of the art system of [6]. Our best average classification accuracy was 86 % using ResNet-50 [25], which is significantly higher than the 75 % of [6].

Figure Analysis. Evaluating figure analysis results is a challenging endeavor as it demands detailed annotation of the figures within research papers. Therefore

Table 1. Figure Classification results across 7 categories. Using CNNs outperforms the previous state of the art approach [6], which used a visual bag-of-words model, by a large margin (86 % vs. 75 %). The last row lists the distribution of data across the categories in our dataset of 60000 samples.

Categories	Graph plots	Flowcharts	Algorithms	Bar plots	Scatter	Tables	Other	**Mean**
[6]	83 %	63 %	73 %	80 %	41 %	64 %	75 %	75 %
AlexNet	82 %	72 %	71 %	85 %	49 %	57 %	**93%**	84 %
ResNet-50	**89 %**	**75 %**	**77 %**	**87 %**	**59 %**	**67 %**	**93 %**	**86 %**
Data stats	20.6 %	12.6 %	6.8 %	6.1 %	2.6 %	2.4 %	48.9 %	14.3 %

most previous works have restricted their evaluation to small datasets or manual inspection [6,20]. The availability of our detailed annotated dataset allows thorough analyses of the various components of our approach. We ran figure analysis experiments on the graph-plot figures from our dataset. The figures were randomly split into two halves for training and testing.

Text Identification. Our figure analysis approach needs access to all the text content within the figures (i.e., axis labels, legend labels, etc.,). Given the extensive progress in the OCR community over the past several decades towards the localization and recognition of text in images and documents, we leveraged state of the art OCR engines (Microsoft OCR [41], Google OCR [42], Abby [43]) for text identification. Figure 5 displays a few results of text localization and recognition using Microsoft OCR [41] on our dataset. While text corresponding to legend labels is often well localized, the text corresponding to axes labels is challenging due to the prevalence of numeric, rotated, sub/superscript, and decimal characters. Our overall accuracy for text recognition was 75.6 % with an F1-score of 60.3 % for text localization. To factor out the effect of OCR errors, we pursued our experiments by using ground truth text-boxes. (See Sect. 7 for results obtained when using text from OCR instead.)

For evaluating axis parsing performance, we independently measured the accuracy of our axes position, axes label, and axes scale detection modules. Axes position (i.e., the plot area extent) accuracy is measured by using the standard bounding box overlap-criteria from object detection [40]. More specifically, we regard a predicted bounding box B_p for the plot-area to be correct if its intersection-over-union with the ground-truth box B_g is above 0.5, i.e., $\frac{B_p \cap B_g}{B_p \cup B_g} > 0.5$. Under this metric, we obtained an accuracy of 99.2 %. For measuring axes label accuracy, we use the same box overlap criteria and obtained an accuracy of 95.9 %. Finally, for evaluating axes scale, we compute the difference (in pixels) between the predicted and ground-truth axes scales, and regard a prediction to be correct if the difference is below a threshold of 5 %. Under this metric, we achieved an accuracy of 91.6 %.

For evaluating legend parsing performance, we independently measured the accuracy of our legend label detection and symbol detection method using the

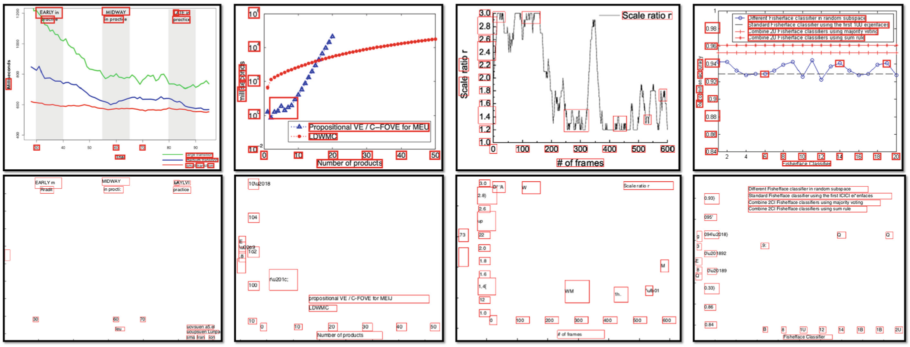


Fig. 5. Scholarly figures present a challenge to state of the art OCR: text localization (top row) and recognition (bottom row) results using [41]. Common errors include (i) missed localizations, e.g., rotated text (left-most, y-axis), numeric text (right-most - ‘2’, ‘4’), (ii) incorrect recognition, e.g., sub/superscripts (left-middle, y-axis), decimals (right-middle - ‘2.2’ as ‘22’), and (iii) false positives, e.g., spurious boxes in plot area.

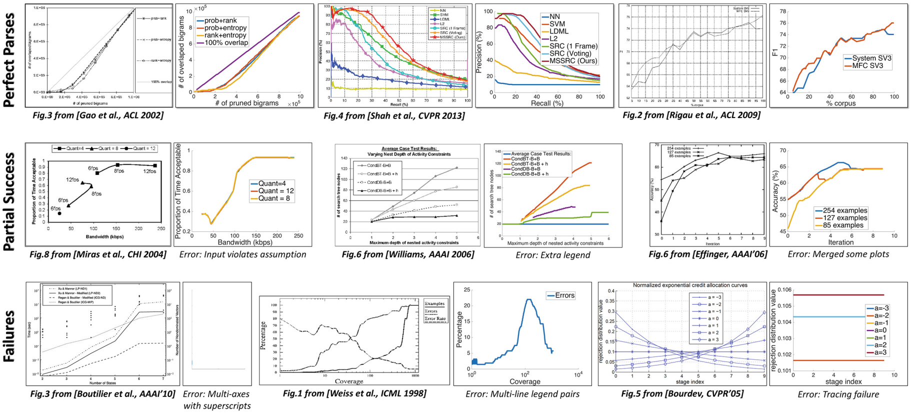


Fig.3 from [Gao et al., ACL 2002]
Fig.4 from [Shah et al., CVPR 2013]
Fig.2 from [Rigau et al., ACL 2009]
Fig.8 from [Miras et al., CHI 2004]
Error: Input violates assumption
Fig.6 from [Williams, AAAI 2006]
Error: Extra legend
Fig.6 from [Effinger, AAAI’06]
Error: Merged some plots
Fig.3 from [Boutilier et al., AAAI’10]
Error: Multi-axes with superscripts
Fig.1 from [Weiss et al., ICML 1998]
Error: Multi-line legend pairs
Fig.5 from [Bourdev, CVPR’05]
Error: Tracing failure

Fig. 6. Qualitative results (Left: original figure, Right: regenerated figure). Top row shows three samples of perfect parses, where our approach understands and regenerates challenging figures. Middle row shows three examples where our parser makes some errors, such as when the input figure violates assumption of being a function, or merges parts of the plots. Bottom row shows failures, such as when figures have multiple y-axis (with superscripts), or have multi-line legends, or have dense plot-data crossings.

box overlap-criteria. Under this criteria, our approach obtained an accuracy of 72.6 % for label detection and 72.7 % for symbol detection.

For evaluating plot-data parsing performance, we used the standard F-measure metric [44] with following statistical definitions: A point $\mathbf{x}_i$ on the predicted path $P_s = \{\mathbf{x}_i\}_{i=1}^{n} = \{(x_i, y_i)\}_{i=1}^{n}$ is counted as true positive if the normalized difference with the ground-truth y'_i is below a threshold, i.e., $(y_i - y'_i) < th$ ($th = 0.02$ in our experiments). A predicted point is counted as false positive if

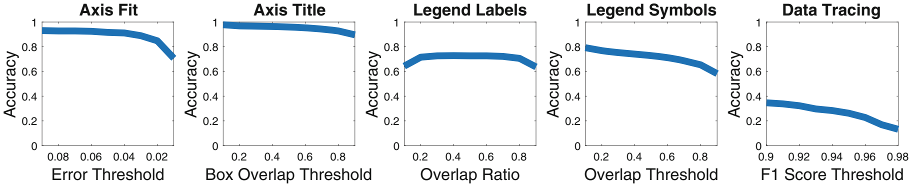


Fig. 7. Evaluation: Our results are robust to the chosen evaluation parameters.

there exists no ground-truth at that position, i.e., $y'_i = \varnothing \cap y_i \neq \varnothing$. Similarly, a false negative is recorded when $y'_i \neq \varnothing \cap y_i = \varnothing$. A predicted point is counted as both false positive and false negative if the predicted value is outside the threshold. With these definitions, we consider a predicted path to be *correct* if its F_1 score is above a threshold Th (95 %). Under this conservative metric, our data-parsing approach achieved an accuracy of 26.4 %. Note that for a figure to be considered correct all the lines must be parsed accurately. We also analyzed the importance of the CNN-based similarity features within our path-finding model. Ignoring these CNN features dropped our accuracy to 23.2 %, confirming their utility.

While the above evaluations reveal the component-level performance, we also evaluated our overall figure analysis performance. Our approach obtained an overall accuracy of 17.3 %. Note that several components need to be sequentially accurate for the entire parsing to be considered correct. Figure 6 displays a few qualitative results obtained using our analysis approach. Our approach does an impressive job despite the high structural variations in the figure as well as the presence of heavy deformations in the plotted data.

Evaluation parameters: To study the sensitivity of the chosen evaluation thresholds within our model towards the final performance, we analyzed our results sweeping over varying parameter settings. As displayed in Fig. 7, the performance is stable across a range of settings.

6 Applications: Query Answering

While our proposed figure parsing approach enables a variety of exciting applications (see Fig. 1), here we describe a functioning prototype of a query-answering system that allows powerful search and querying of complex figure content across multiple papers. The input to our query-answering system is a templated (textual or numerical) query that requests rich semantic details about a specific dataset. For example, *Best method on the LFW dataset?*, *Best precision at 0.3 recall on BSDS dataset?*, etc. The output is a textual response (numerical or string value) obtained by analyzing the parsed content of all figures that match the requested dataset. We assume a simple query representation with a structured template that has two parts: the dataset (e.g., 'PASCAL VOC detection', 'UCI

IRIS classification', etc.,) and the metric (e.g., 'precision vs. recall', 'accuracy vs. #dimensions', etc.,).

Table 2. Quantitative results on complex query-answering. 'Top-5' indicates the results obtained when the predicted answer is within the top 5 answers.

Approach	Textual (top-1)	Textual (top-5)	Numerical
Baseline	25.3 %	62.3 %	-
Color-only	38.7 %	64.6 %	32.3 %
Ours	**47.7 %**	**72.6 %**	**45.3 %**

Given a specific query, we first retrieve all relevant figures (across multiple papers) from our corpus that match the requested dataset and metric by searching the figure meta-data (captions). The retrieved figures are then processed using our approach and the parsed content is then collected into a simple data-table representation. Finally, the query is run through the collected data and the requested quantity is extracted. We ran experiments on a collection of over 3,500 textual and numerical queries. The textual queries are formatted such that they request the specific *label* (amongst those indicated by the legend labels) that is the *best* (in terms of the y-axis values) either at specific points of the domain (i.e., x-axis values) or the overall domain. Similarly, the numerical queries are formatted such that they request the best y-axis value obtained at specific points or overall domain (x-axis). (Please see supplementary for more details). Queries were evaluated by comparing the predicted response to the ground-truth.

Table 2 summarizes the results obtained using our approach. Numerical queries are judged correct if the returned value is within 2 % of the correct value. We compare our results to (i) a baseline method that naively picks a response to a query without parsing the plotted data, i.e., by randomly picking one of the classified legend labels; and (ii) a version of our approach that only uses the color-feature representation. Our approach obtains impressive results, thereby reaffirming the potential value it unleashes. Figure 8 displays qualitative results on some examples of queries possible using our answering system.

While we have presented the query-answering system as a proof-of-concept, we highlight a few other exciting potential applications:

Figure captioning: In the pursuit of immediate dissemination of results, authors often miss providing meaningful captions to their figures in papers [45,46]. Our proposed figure parsing approach can help towards automatic caption generation. Our parsed structural representation can be used to not only create simple-templated captions [47], but also help generate complex summaries [48,49].

Accessibility: Developing interfaces that can provide simple and convenient access to complex information has huge benefits across multiple domains [50,51]. While authors often summarize interesting observations about their figure content in

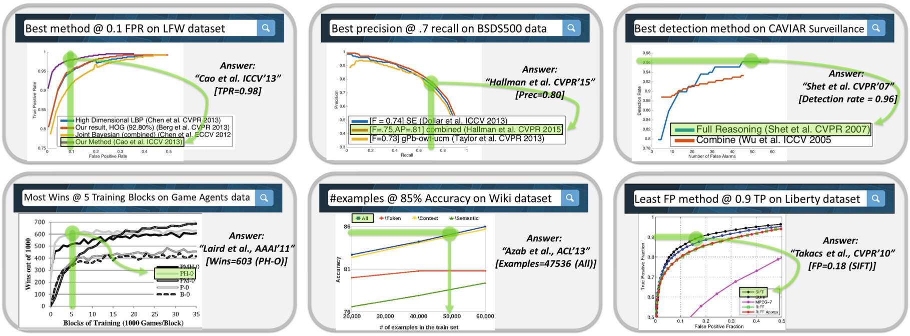


Fig. 8. Qualitative results demonstrating the utility of our parsing approach towards complex query-answering. Our approach is not only able to successfully parse challenging figures, but also answer interesting queries by summarizing results across multiple papers. Queries in the top row collate plot-data from multiple papers – for e.g., in case of LFW (top-left), our method combined results from 4 different papers: *Cao ICCV'13, Chen CVPR'13, Berg CVPR'13*, and *Chen ECCV'12* to answer the query.

their paper text, the alignment between the figure elements and their corresponding mentions in the body text is currently unavailable [52]. Our figure parsing approach can help towards bridging this gap, and thereby facilitates the development of richer visualization interfaces [53,54].

Plagiarism detection: Recent years have witnessed a surge in papers reproducing hitherto published results [55]. Identifying such plagiarized articles is of utmost concern to academic committees and publishers [56]. Our figure parser can help towards their detection by analyzing and matching their result-figure contents.

7 Discussion

With scores of papers being published every year, it is imperative to devise powerful academic search systems that can discover important papers and identify influential citations, thereby alleviating researchers from the enormous information overload. In this paper, we introduced FigureSeer, an end-to-end framework for parsing figures within research papers that enables rich indexing and search of complex result content. We have presented a novel approach for result-figure parsing that not only handles the problems with clutter and deformations of the plotted data, but is also robust to the variations in the format and design of figures. Our experimental analysis has confirmed that figure parsing in scholarly big data is a challenging vision application. We hope our work will spur further exciting research in this domain.

While our current framework is generalizable for parsing a variety of result-figures, it has only scratched the surface with interesting open challenges ahead. OCR is a critical component towards the success of our framework. State-of-the-art and commercial OCR engines have limited success in case of scholarly

Table 3. Results obtained using OCR-based [41] text identification. (Query-answering QA shows top-1, top-5, and numerical results.) Poor OCR performance hurts the different components of our framework.

Task	Result using OCR-text	Result using text-boxes
Axes position	79.4 %	99.2 %
Axes label	42.5 %	95.9 %
Axes scale	60.5 %	91.6 %
Legend label	41.6 %	72.6 %
Legend symbol	63.2 %	72.7 %
Plot-data parsing	21.4 %	26.4 %
Overall	7.2 %	17.3 %
QA (Ours)	19.2 %, 33.7 %, 17.8 %	47.7 %, 72.6 %, 45.3 %
QA (Baseline)	10.8 %, 32.0 %, -	25.3 %, 62.3 %, -

figures. Table 3 reports results obtained by our framework when using text from OCR. Our preliminary attempts at post-processing the OCR results with deep learning based reasoning only partially redressed these errors. Improving OCR performance by addressing the challenges posed within scholarly figures is an interesting and open future endeavor.

Our plot-data parser currently suffers from successfully parsing the plotted data in presence of heavy clutter (see Fig. 6, bottom right). Techniques from vascular tracking such as [57] could be applicable here. Our legend parser currently cannot handle labels spanning multiple lines. Our axes parser assumes the axes scale are either linear or logarithmic, with their tick labels being limited to numeric values. Finally, our figure analysis approach currently models and trains the different components (axes, legend, and plot-data parser) independently. Jointly modeling all the components and training them together within an end-to-end deep network is an exciting endeavor.

Acknowledgments. This work was in part supported by ONR N00014-13-1-0720, NSF IIS-1338054, and an Allen Distinguished Investigator Award. We thank Isaac Cowhey, Rodney Kinney, Christopher Clark, Eric Kolve, and Jake Mannix for their help in this work.

References

1. Khabsa, M., Giles, C.L.: The number of scholarly documents on the public web. PLoS ONE **9**(5), e93949 (2014)
2. ArXiv stats. http://arxiv.org/stats/monthly_submissions
3. Tufte, E.R.: Visual display of quantitative information. In: Graphics Press, Cheshire (1983)
4. Grice, P.: Logic and conversation. In: Speech Acts (1975)

5. Heer, J., et al.: Crowdsourcing graphical perception: using mechanical turk to assess visualization design. In: CHI (2010)
6. Savva, M., et al.: ReVision: automated classification, analysis and redesign of chart images. In: UIST (2011)
7. Instructions instructions (Using Color). In: AAAI (2016). http://www.aaai.org/Publications/Templates/AuthorKit.zip
8. Instructions for ACL proceedings (Section 3.8). In: ACL (2015). http://www.acl2015.org/files/acl2015.pdf
9. Apache PDFBox. https://pdfbox.apache.org
10. Choudhury, S.R., et al.: Automatic extraction of figures from scholarly documents. In: DocEng (2015)
11. Clark, C., Divvala, S.: Looking beyond text: extracting figures, tables, and captions from computer science paper. In: AAAI Workshop (2015)
12. Kuhn, T., et al.: Finding and accessing diagrams in biomedical publications. In: AMIA (2012)
13. Choudhury, S.R., Giles, C.L.: An architecture for information extraction from figures in digital libraries. In: WWW (Companion Volume) (2015)
14. Chan, J., et al.: Searching off-line arabic documents. In: CVPR (2006)
15. Liu, Y., et al.: Tableseer: automatic table metadata extraction and searching in digital libraries. In: JCDL (2007)
16. Kae, A., et al.: Improving state-of-the-art OCR through high-precision document-specific modeling. In: CVPR (2010)
17. Wu, J., et al.: CiteseerX: AI in a digital library search engine. In: AAAI (2014)
18. WebPlotDigitizer. http://arohatgi.info/WebPlotDigitizer
19. Im2Graph. http://im2graph.co.il/
20. Wu, P., Carberry, S., Elzer, S., Chester, D.: Recognizing the intended message of line graphs. In: Goel, A.K., Jamnik, M., Narayanan, N.H. (eds.) Diagrams 2010. LNCS (LNAI), vol. 6170, pp. 220–234. Springer, Heidelberg (2010). doi:10.1007/978-3-642-14600-8_21
21. Xu, S., McCusker, J., Krauthammer, M.: Yale image finder (YIF): a new search engine for retrieving biomedical images. Bioinformatics **24**(17), 1968–1970 (2008)
22. Choudhury, S., et al.: A figure search engine architecture for a chemistry digital library. In: JCDL (2013)
23. Li, Z., et al.: Towards retrieving relevant information graphics. In: SIGIR (2013)
24. Krizhevsky, A., et al.: Imagenet classification with deep convolutional neural networks. In: NIPS (2012)
25. He, K., et al.: Deep residual learning for image recognition. In: CVPR (2016)
26. Deng, J., et al.: ImageNet: a large-scale hierarchical image database. In: CVPR (2009)
27. McCullagh, P., Nelder, J.: Generalized linear models. In: Chapman and Hall, London (1989)
28. Breiman, L.: Random forests. In: Machine Learning (2001)
29. Felzenszwalb, P., Veksler, O.: Tiered scene labeling with dynamic programming. In: CVPR (2010)
30. Joachims, T.: Training linear svms in linear time. In: KDD (2006)
31. Felzenszwalb, P., et al.: Discriminatively trained, multiscale, deformable part model. In: CVPR (2008)
32. Zagoruyko, S., Komodakis, N.: Learning to compare image patches via cnns. In: CVPR (2015)
33. Han, X., et al.: MatchNet: unifying feature and metric learning for patch-based matching. In: CVPR (2015)

34. Hadsell, R., et al.: Dimensionality reduction by learning an invariant mapping. In: CVPR (2006)
35. Shechtman, E., Irani, M.: Matching local self-similarities across images and videos. In: CVPR (2007)
36. Dillencourt, M.B., Samet, H., Tamminen, M.: A general approach to connected-component labeling for arbitrary image representations. J. ACM (JACM) **39**(2), 253–280 (1992)
37. Jia, Y., et al.: Caffe: convolutional architecture for fast feature embedding (2014). arXiv:1408.5093
38. Semantic scholar. https://www.semanticscholar.org/
39. Sorokin, A., Forsyth, D.: Utility data annotation with amazon mechanical turk. In: CVPR Workshop (2008)
40. Everingham, M., et al.: The PASCAL visual object classes (VOC) challenge - a retrospective. In: IJCV (2015)
41. Microsoft project oxford. https://www.projectoxford.ai/vision
42. Smith, R.: An overview of the tesseract OCR engine. https://github.com/tesseract-ocr/tesseract
43. ABBYY Finereader 9.0. http://www.abbyy.com
44. Hou, X., Yuille, A., Koch, C.: Boundary detection benchmarking: beyond F-measures. In: CVPR (2013)
45. Corio, M., et al.: Generation of texts for information graphics. In: EWNLG (1999)
46. Carberry, S., et al.: Extending document summarization to information graphics. In: ACL Workshop (2004)
47. Kulkarni, G., et al.: Baby talk: understanding and generating simple image descriptions. In: CVPR (2011)
48. Moraes, P., et al.: Generating summaries of line graphs. In: INLG (2014)
49. Chen, X., Zitnick, C.: A recurrent visual representation for image caption generation. In: CVPR (2015)
50. Ladner, R.: My path to becoming an accessibility researcher. In: SIGACCESS (2014)
51. Russell, B.C., et al.: 3D Wikipedia: using online text to automatically label and navigate reconstructed geometry. In: Siggraph Asia (2013)
52. Seo, M.J., et al.: Diagram understanding in geometry questions. In: AAAI (2014)
53. eLife Lens: lens.elifesciences.org
54. Tableau software. http://www.tableau.com/
55. Williams, K., et al.: Simseerx: a similar document search engine. In: DocEng (2014)
56. Noorden, V.: Publishers withdraw more than 120 gibberish papers. In: Nature (2014)
57. Sironi, A., et al.: Multiscale centerline detection by learning a scale-space distance transform. In: CVPR (2014)