PhyloParser: A Hybrid Algorithm for Extracting Phylogenies from Dendrograms

Po-shen Lee*, Sean T. Yang*, Jevin D. West†, Bill Howe†
Department of Electrical Engineering*, Information School†
University of Washington, Seattle, Washington 98115
Email: {sephon, tyyang38, jevinw}@uw.edu and billhowe@cs.washington.edu

***Abstract*—We consider a new approach to extracting information from dendograms in the biological literature representing phylogenetic trees. Existing algorithmic approaches to extract these relationships rely on tracing tree contours and are very sensitive to image quality issues, but manual approaches require significant human effort and cannot be used at scale. We introduce PhyloParser, a fully automated, end-to-end system for automatically extracting species relationships from phylogenetic tree diagrams using a multi-modal approach to digest diverse tree styles. Our approach automatically identifies phylogenetic tree figures in the scientific literature, extracts the key components of tree structure, reconstructs the tree, and recovers the species relationships. We use multiple methods to extract tree components with high recall, then filter false positives by applying topological heuristics about how these components fit together. We present an evaluation on a real-world dataset to quantitatively and qualitatively demonstrate the efficacy of our approach. Our classifier achieves 89% recall and 99% precision, with a low average error rate relative to previous approaches. We aim to use PhyloParser to build a linked, open, comprehensive database of phylogenetic information that covers the historical literature as well as current data, and then use this resource to identify areas of disagreement and poor coverage in the biological literature.**

Keywords*—*Viziometrics, Phylogenetic Trees, Deep Learning, Convolutional Neural Networks.

I. Introduction

Scientific results in the biomedical literature are frequently presented visually with figures, diagrams, and tables, but the information contained in these objects are inaccessible to text-oriented computational approaches. As part of the viziometrics project, we are working to develop a general framework for information extraction from the visual literature using computer vision and machine learning approaches.

In this paper, we focus on extracting information from phylogenetic trees. These trees are used extensively within genetics, cladistics, conservation biology, medicine, public health and many other areas of biology [1] to organize evolutionary relationships between species into a hierarchy rendered as a dendrogram (Figure 6). They are used to track the evolution and spread of viral infections [2], migration of species [3], and for comparing genetic sequences [4].

Public repositories for phylogenetic information have been created including TreeBASE [5] and MorphoBank [6]. These databases are intended to organize and aggregate results to help build scientific consensus ahout the tree of life [7]. However, these databases are relatively new, and are not comprehensive for at least two reasons: Results from older papers are missing entirely, and even among current papers, there is no mandate to use these repositories. In 2017, there are more then 40 thousands phylogenetic trees available on PMC, which is three times the number of trees available on TreeBASE. More broadly, policies designed to encourage researchers to clean and share their data have had limited success [8]. We aim to use PhyloParser to construct a new database of phylogenetic information, with goals similar to that of TreeBASE, but to derive it automatically from the scientific literature itself to increase coverage and reduce human effort.

Previous approaches to this problem either rely on human input or are sensitive to noise, making them inadequate for our purposes. For example, the interactive approach proposed by Laubach et al. [9] would require hundreds of hours to process typical datasets. Previous automated approaches rely on line-tracing techniques that are sensitive to noise. The figures in the literature are extremely heterogeneous: they may involve complex annotations and background formatting, vary in size and resolution, and use inconsistent spacing between lines, text, and other elements. These complications prevent any one method from being successful in all cases. Our approach, in contrast, is to combine machine vision approaches with a topological “grammar” of how these trees are constructed in order to significantly reduce errors.

We automatically recognize the fundamental components of dendrograms: horizontal branches, vertical branches and the text of species names. We then use hough transform and convolutions to extract these components with high recall, then filter false positives by applying topological heuristics about how these components fit together. The tree structure can be recovered by assembling these components both both top-down and bottom-up. This approach allows us to ignore noise in pixel level and enables partial recovery of a dendrogram that may have poor global quality but good local quality; we find this trait to be critical in extracting information from characteristically noisy images in the literature. For instance, Hughes et al. used a dataset selected to favor their approach [10]; even on this curated dataset, our approach extracts 11% more perfectly recovered instances, approximately 80% information from the imperfect cases, and runs significantly faster.

The errors made by our algorithm tend to occur when the resolution and the quality of the images are low, when the text and the lines overlap, and when tree branches are of a color very close to the background color. None of these sources of error appear insurmountable.

In this paper, we make the following contributions.

- We present a classification approach to recognize phylogenetic tree diagrams in the literature using deep neural networks.
- We present a hybrid algorithm for parsing dendograms that recognizes the basic elements of the diagram and then assembles them topologically to be more resistant to local noise.
- We evaluate these methods on datasets from prior work as well as a new dataset extracted from one million papers in PubMed.
- We organize these methods into an end-to-end system called PhyloParser that produces new relationships in a machine-readable format.
- PhyloParser and all datasets we use in our experiments are available online to enable future research.

This paper is organized as follows. In Section II, we describe related work in figure classification and information extraction. In Section III, we describe the algorithms in detail. In Section IV, we evaluate PhyloParser on multiple datasets and discuss the results. We conclude in Section V.

II. Related Work

Lu et al. proposed an automated algorithm for extracting components form 2-D line curves [11]. They report a match rate of 72.5% by manually compare 40 plots and the corresponding redrawn plots. Kataria et al. extended the algorithm to handle scatter and curve-fitted plots and also proposed a text block detection algorithm to parse tick labels and legends [12]. These early research introduces computer vision techniques for parsing artificial images; however, no results on the accuracy of recovering raw data from the figures was included.

Savva et al. proposed a classifier for identifying 10-category chart image and an algorithm to extract data from bar charts and pie charts [13]. Their method achieved impressive result but it highly depends on manually locating text and limits to plots with colors. In 2016, Siegel et al. proposed a thorough pipeline for extracting data from curve plots [14]. Their pipeline integrates several modules such as axes parser, legend parser, curve parser, etc. For the data extraction part, they trained a siamese network to extract similarity features of curves and then use rank SVM formulation with boosting to find best path of each curve. Their approach achieve an overall accuracy of 17.3% because several modules all need to be accurate for the entire parsing to be considered perfect.

These projects focused on parsing plots that present quantitative data; parsing diagrams are increasingly recognized to be even more challenging. In 2016 Kembhavi et al. proposed Diagram Parse Graphs, a first model parsing and studying semantic interpretation of diagrams that illustrates relationships between nature objects [15]. The two related works published in 2016 reveal the increasing attention of understanding visual illustration in the computer vision community. Compared to natural images, visual illustrations encode rich knowledge that has been well organized, which are potentially sources for artificial intelligence studies.

A. Rambaut proposed the first interactive tool to convert a tree image into machine-readable format [16]. It requires the user to reconstruct the tree by clicking on each of its nodes in turn. In 2007, Laubach et al. proposed TreeSnatcher, a semi-automatic application to recover the phylogenetic data under the user's supervising in GUI [9]. Later in 2012, they upgraded the application with more interactive tools of image processing and drawing function [17]. The semi-automatic application effectively reduces the consuming time for accurately parsing a tree. However, it is not a solution for parsing a large collection. In 2011, Hughes proposed TreeRipper, an web application automatically convert the tree image into NEXUS, Newick and phyloXML formats [10]. The application aims at both bifurcating and multifurcating trees, but it has strict prerequisites for the style of input images. The algorithm starts with a sequence of heuristic cleaning steps followed by tracing the contour of the tree topology. The author reports 37 successful samples out of 114 phylogenetic tree diagrams. As the pioneer of parsing tree, TreeRipper shares the same goal with our study; however, it is not tolerant to uncleaned noise, which is very crucial to the contour tracing and restricts the accuracy of TreeRipper.

For the classification tasks, convolutional neural networks have been shown to be effective for natural images. In recent study, Noah Siegel et. al fine-tuned the pretrained AlexNet [18] and ResNet-50 [19] with 60,000 figures annotated using Amazon mechanical turk [14]. They created seven categories and focused on classifying plot graphs such as bar plots, line plots, etc. They obtained 84% and 86% accuracy from AlexNet and ResNet-50 respectively. In contrast, we obtained 95% accuracy from our own dataset categorized into eight figure types using AlexNet trained from scratch.

III. PhyloParser Algorithms and Architecture

PhyloParser includes a classifier for identifying tree diagram (Section III-A) and a tree parser (Section III-B) for recovering phylogenies from a tree diagram. We use Convolutional Neural Networks (CNNs) to train figure type classifiers. Our tree parser is designed for trees with following prerequisites, which appear to be met in practice:

- The tree is constructed by horizontal and vertical lines.
- The root is on the left and the leaves are on the right.
- The background color is lighter then the tree structure and the text.

The algorithm starts from collecting key components of a tree diagram and then assembles these components to recover the tree structure.

A. Identifying Phylogenetic Trees

To identify phylogenetic trees, we use a neural network trained on our own dataset to classify images into one of eight categories: equations, photos, diagrams, tables, plots, electrophoresis gels, metabolic pathways and phylogenetic trees. We describe this dataset in more detail in Section IV-A. We train the network using the Caffe framework [20] installed on an Amazon EC2 instance (g2.2xlarge). We reserved 80% images for training, 10% images for validation in training phase, and 10% images for testing. Iteration stops when the accuracy of testing on validation data is steady and invariant to decreasing learning rate. Experimental result is described in Section IV-B.

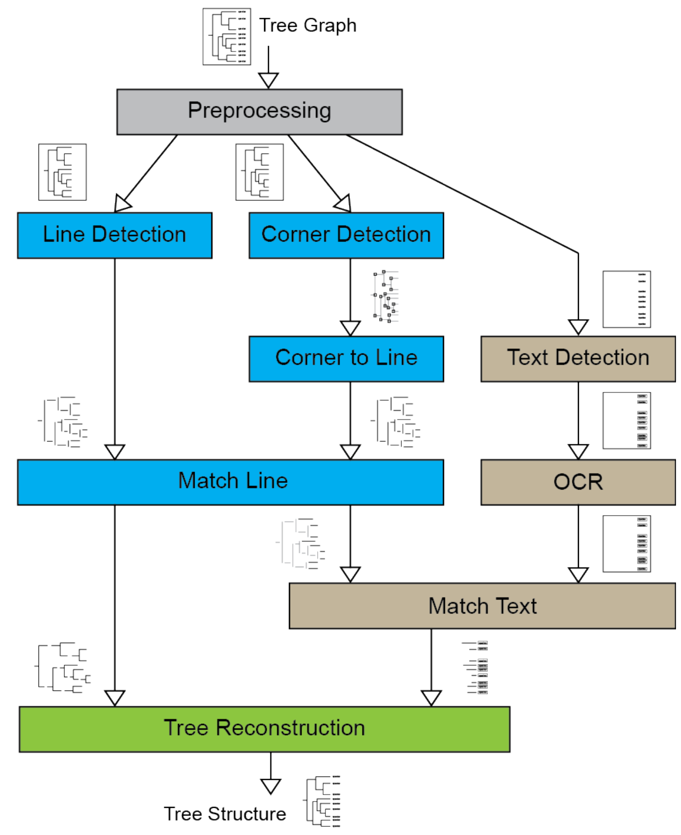

Fig. 1. Pipeline of parsing a tree diagram. There are four stages: preprocssing, tree component extraction, text extraction, and tree reconstruction. First, we remove color background and separate text from tree diagram (grey box). Next, we extract tree components (blue boxes) and specie names (pink boxes) from the image. Finally, we connect the components to recover the tree structure (green box).

B. *Parsing Phylogenetic Trees*

Figure 1 illustrates the pipeline of our parsing algorithm. We divide the algorithm into four stages: preprocessing (grey), tree component extraction (blue), text extraction (pink), and tree reconstruction (green). During preprocessing, we remove any color background if it exists, then separate the tree structure from the text. During tree component extraction, we recognize tree components including vertical lines, horizontal lines, corners, and joints. Then we reorganize the components based on their connectivity. During the text extraction, we locate text regions and convert them into text using standard optical character recognition (OCR) to acquire species names. In addition, we associate each species to the corresponding tree leaf. During tree reconstruction, we assemble all components together to recover the entire tree. In the following sections, we will describe each stage.

1) Preprocessing: Our parsing algorithm is based on identifying horizontal and vertical line segments locally, then topologically connecting these segments into a global tree. We find that high-accuracy line detection at this stage is critical for achieving a low-error result.

To make our algorithm scale invariant, we first resize the input image to the dimension in which the line thickness is normalized to be lower then four pixels. Without this step, thick lines can be mistaken for dark patches that will be removed in future steps. The line thickness is determined by iteratively applying a morphological open operation: an erosion to reduce thickness followed by a dilation to expand thickness, which can "open" connections between elements. For each iteration, we increase the morphological kernel size. The iteration terminates when the sum of pixel values in the image is half of the sum of pixel values acquired from the original image (indicating the tree structure has been completely erased), or stops after 15 iterations. Too many iterations can make it difficult to distinguish the lines from a dark background. In this case, we do not resize the image.

A common source of errors for line detection is text, so after the opening process, we need to separate text from the tree structure. We acquire contours by using the simple contour finder in OpenCV, then use the heuristic that the tree structure is typically the largest contour in an image. This largest such contour is used to build a *tree mask*, and the remaining contours are used to build a non-tree mask. We apply this tree mask to distinguish the main structure of the tree from other sources of lines, including text.

Next, we remove color patches in the background. These patches decrease the gradient of tree boundaries, making it difficult for the line detector to identify segments reliably. To detect color patches, we use canny edge detection to highlight all boundaries, followed by a morphological close operation with a 5×5 rectangular kernel to backfill the tree structure. We locate the pixels within these boundaries to identify color patches. Next we extract the colors from these background pixels and whiten all pixels with the same colors in the original image.

Finally, we use bilateral filter to sharpen edges, which improves line detection specifically for thin lines.

2) Tree Component Extraction: The main task in this stage is to extract vertical lines and horizontal lines. We first use the non-tree mask to whiten irrelevant pixels in the image and then detect lines using two approaches: Hough transform [21] and endpoint detection.

Hough transform can detect lines in given degree. We only need vertical lines and horizontal lines. The text pixels that are not successfully excluded by the mask can produce many short lines. We identify and drop each short vertical line l_{ver} and each short horizontal line l_{hor} according to the following two heuristics:

$$\texttt{drop}\ l_{ver}\ \texttt{if}\ length(l_{ver}) < 8 + height(image)/100$$

$$\texttt{drop}\ l_{hor}\ \texttt{if}\ length(l_{hor}) < 3 + width(image)/100$$

We determined the threshold values 3 and 8 experimentally. We set a higher threshold for vertical line because the minimum length of true vertical lines are usually longer then the true horizontal lines in tree topology. We determined that our experimental results are not sensitive to this parameter in the range 6 to 15 pixels. This filtering step does remove a portion of true lines and thus prune a tree. In the later stage, we will recover the missing part via bottom-up reconstruction methods.

A Hough-transform-based line detection, such as the one we use here, does not guarantee a 100% recall. To increase

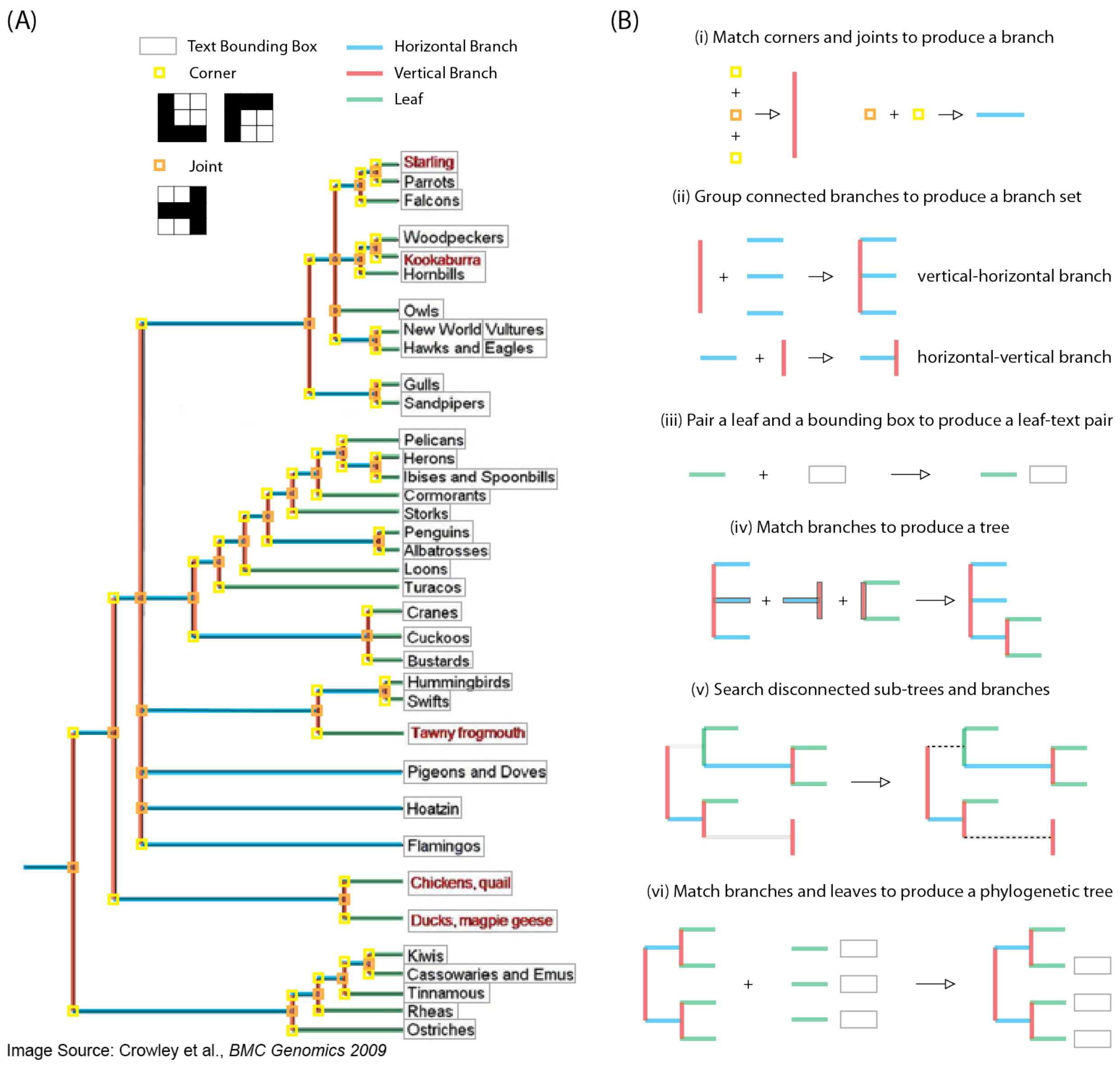

Fig. 2. (A) Tree components and (B) reconstruction logic. Our parsing algorithm is based on accurate line detection. Beside using classical Hough transform to detect lines, we also extract lines by detecting corners and joints as shown in part (i) of (B). Texts are located by contour finder and converted by Google Tesseract. In (B) We visualize the concept of "Match Line" in part (ii), "Match Text" in part (iii) and "Tree Reconstruction" in part (iv) to (vi).

the recall, we capture the corners and joints (see Figure 2(A)) and pair them to acquire vertical lines and horizontal lines. We first binarize the image with a threshold of 180. We use simple convolution with the three masks shown in Figure 2(A) to expose top corners, bottom corners, and joints, refined by two thresholds: (1) a high-pass threshold to include highly responsive pixels and (2) a low-pass threshold to eliminate highly responsive pixels that are entirely surrounded by black pixels. Next, we pair top corners with bottom corners to create vertical lines as well as corners with joints to create horizontal lines (Figure 2(B)(i)).

Since multiple lines can be detected from a single thick line, we group those lines that come from the same source to avoid duplicate pairs in the next step. Figure 2(A) shows the ideal result of line detection and corner detection. Next, we connect vertical lines and horizontal lines to create branch sets (Figure 2(B)(ii)). For each vertical line, we associate it with right connected horizontal lines as a vertical-horizontal branch (v-h-branch). For each horizontal line, we associate it with the right connected vertical line as a horizontal-vertical branch (h-v-branch). These branch sets will be used in the tree reconstruction process. The horizontal lines that are included in v-h-branches but have no right connected vertical lines are defined as "leaves," which will be used to connect species names in the text extraction stage. Since the leaves can generated from the remaining text, we further use a binary classifier to verify the leaves. The features selected are based on the assumption that a leaf typically has an endpoint on the right followed by text. As an example of how these features manifest in practice, consider Figure 3. We use the pixel values in the red area as the image feature, together with the horizontal distance between the right endpoint of a leaf

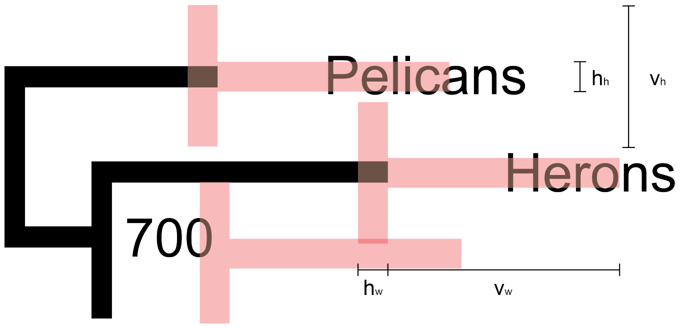

Fig. 3. To identify leaves, we train a model based on spatial features between lines and text. We design the features to capture the unique pattern of leaves: a vertex followed by text. We extract the raw pixels (colored red), along with the horizontal distance between the right endpoint of a leaf and the mean x-coordinate of the endpoints of all leaves. To determine these features, we set $h_w = 3$, $h_h = 3$, $v_w = 15$, and $v_h = 9$.

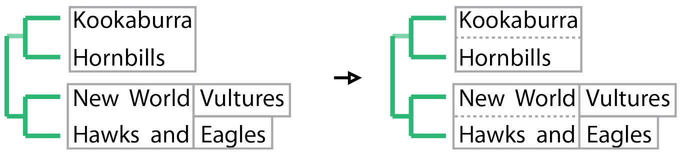

Fig. 4. Method of separating cross-line bonding box. Text in different lines can be bonded together if they are very close. A reasonable segmentation can be deducted from (1) other bonding boxes or (2) corresponding leaves. We seek the first solution prior to the second solution, because the perfect segmentation from the second solution only applicable when all corresponding leaves are found.

and the mean position of the right endpoints of all leaves. We compile a training set containing 3368 positive examples and 1201 negative examples, extracted from 100 tree diagrams (not included in the test set). We tested several classifiers and finally chose random forest, giving the highest accuracy of 93.4%. Most of the false leaves are produced from text that are extremely close to the vertical lines. The classifier is effective to identify these "fake" leaves.

3) Text Extraction: We use the tree mask to remove the tree diagram as the first step of text detection. Second, we again use the contour finder to segment characters, strings, and any other irrelevant items such as arrows, numbers, etc. For each contour, we generate a bonding box for further use. Here, we eliminate tall and thin bounding boxes that are unlikely to contain text. We define these boxes as those satisfying two conditions: (1) the aspect ratio (height / width) is greater than 10 and (2) the height is also greater than 10.

Third, we generate leaf-text pairs (Figure 2(B)(iii)). For each leaf, we associate it with the bounding boxes locating right to the leaf. For the case that a bonding box links to multiple leaves, we divide the bounding box in the following methods: (1) Divide the bounding box based on nearby bounding boxes that suggests a reasonable division (see lower leaves in Figure 4). In this example, we separate "New World" from "Hawks and" based on the known patterns "Vultures" and "Eagles". (2) For the case without such hint (see upper leaves in Figure 4), we divide the bounding box at $(y^i_{leaf} + y^{(i+1)}_{leaf})/2$, where i denotes the associated leaf and y denotes the corresponding location on y-axis. We found approximately 50% of our test images need such process to separate at least one

TABLE I. EVALUATION OF FIGURE-TYPE CLASSIFIER USING HOLD-OUT TESTING SET WITH 1878 IMAGES.

Figure Type	**Precision / Recall**		
	AlexNet	**ResNet-50**	**Bag Of Feature** [13], [23]
Equation	98% / 97%	97% / 97%	97% / 97%
Photo	95% / 100%	95% / 96%	93% / 95%
Electrophoresis Gels	89% / 98%	96% / 97%	85% / 80%
Plot	98% / 95%	94% / 94%	90% / 91%
Table	100% / 97%	95% / 94%	95% / 94%
Diagram	88% / 92%	74% / 74%	75% / 74%
Metabolic Pathway	94% / 84%	84% / 76%	73% / 77%
Phylogenetic Tree	**99% / 89%**	**87% / 93%**	**91% / 86%**
Accuracy	**95%**	90%	88%

bounding box.

Text in bounding boxes are converted using standard OCR methods [22]. For the leave not associated with any bounding box, we scan the area to the right with 10-pixel-high box to seek missing text. The text successfully associated with leaves will be used as the species names in the next stage, while the text not pairing with any leaf (an *orphan* text box) will be used to identify missing leaves.

4) Reconstruction: We reconstruct the tree by connecting h-v-branches and h-v-branches with their common lines. However, this step does not guarantee a perfect reconstruction because some vertical lines and horizontal lines are likely missing in the step of detection. In this case, we obtain several sub-trees. Each of these sub-trees will either be missing a horizontal line at its root, or will have at least one broken branch. A broken branch is a vertical line not connected to at least two leaves on its right. To overcome this issue, we develop two methods to search for the missing connection:

A. Search for any existed sub-trees or vertical branch in the right area of the broken branch.
B. Search for any orphan text boxes in the right area of the broken branch.

Figure 2(B)(v) shows an example in which three sub-trees are not linkable because of undetected lines (highlighted in light grey). We reconnect the upper sub-tree using method A and the lower vertical branch using method using method B. Method C handles a particular tree style that a tree does not use horizontal lines to tip species, for instance the lower part in Figure 2(B)(v). For a broken branch, we associate it with the orphan text boxes within a distance of 15 pixels, defined by observation. Method C does not handle a rare case that horizontal lines are used for both top and bottom leaves but not for the middle leaves in a multifurcating tree.

The final step of tree reconstruction is merging the recovered tree structure and species names (embedded in leaf-text pairs and orphan text boxes). We have associated the text with the leaves or the vertical branches in the previous steps, so we

TABLE II. EVALUATION OF THE TREE PARSER USING TREERIPPER DATASET (100 IMAGES) AND OUR OWN DATASET (141 IMAGES).

Figure Type	Error Rate	Count of Perfect Extraction
PhyloParser Dataset		
Base	0.238	39
Base + A	0.156	**54**
Base + A + B	**0.148**	51
TreeRipper Dataset [10]		
TreeRipper	N/A	37
Base + A	**0.116**	**48**
Base + A + B	0.120	44

only need to traverse the tree structure to produce the final tree string in the Newick format.

IV. Evaluation

A. *VizioMetrics Dataset*

In our previous work [23], we categorized the figures into five main types: equation, photo, diagram, table, and plot. To access phylogenetic trees (as a sub-type of diagram), we separated them from diagrams as an independent category. We also separate metabolic pathways from diagrams and electrophoresis gels from photos for the other research projects.

We use our online labelling tool [24] to create a training corpus. The tool allows users to (1) collect figures with the same figure type and (2) collect figures with given keywords in captions. In the first mode, the tool returns a set of figures that are classified as the same figure type by our first-generation classifier [23] and asks users to label true positive images. The tool changes the returned figure type in each round and stack the unlabelled figures into the next round. These figures will eventually be labelled with their true types in future rounds. We use the first mode to collect more equations, photos, diagrams, tables, and plots. In the second mode, the users can create any category and search for candidate images based on captions. We use this mode to collect phylogenetic trees, metabolic pathways, and electrophoresis gels. We labelled 15,507 images from the tool together with 3,271 labelled images used in [24]. Finally we collected 18,778 images: 1411 electrophoresis gels, 1871 equations, 1119 metabolic pathways, 3347 photos, 1308 phylogenetic trees, 2849 diagrams, 2193 tables and 4680 plots. We use this image set to train new classifiers using CNNs.

To evaluate our tree parser, we create a test image set randomly collected from the 1308 phylogenetic trees. The test image set includes only trees constructed by horizontal lines and vertical lines, i.e. circular trees and cladograms are not included. In addition, we do not include low-resolution trees with any ambiguous branches and species names that are not readable by human. Despite that our approach is able to recover these trees partially, we are not able to create an absolutely correct ground-truth for a fair comparison. Finally, we compiled a test image set consisting of 141 phylogenetic tree images.

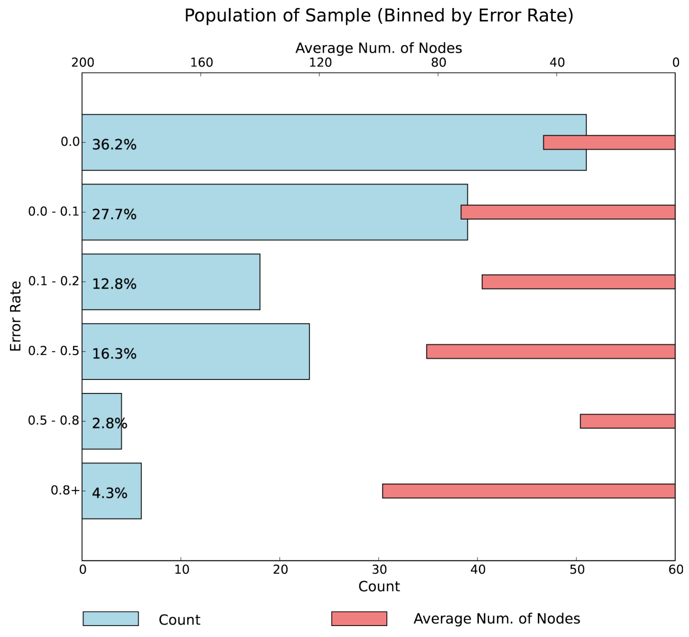

Fig. 5. Histogram of reconstructed trees categorized by error rates. The wide bars show the figure counts and the thin bars show the average number of nodes. We obtain 51 perfectly reconstructed trees and 108 reconstructed trees with error rates below 0.2. The performance of our algorithm degrades when the size of tree increases.

B. *Figure Classification*

We reserved 80% images for training, 10% images for validation in training phase, and 10% images for testing. We tested two methods for training the networks: (1) train the networks from scratch, and (2) fine-tune the networks were pretrained on the 1.2 million images from ImageNet [18]. We received very similar results from the two methods so we just reported the one with better performance for each architecture (AlexNet trained from the scratch data and fine-tuned ResNet-50) in Table I.

Different from [14], we obtained the highest accuracy from AlexNet trained from scratch. In general, classifying diagrams and the corresponding sub-types are more difficult then other types of figures due to the higher diversity, showing in all the tree models. In details of identifying phylogenetic trees, we obtained better precision from AlexNet but better recall from ResNet-50. Diagrams are the major false positives and false negatives for the both models. The better f1-score is achieved by AlexNet (0.94 vs 0.90.)

C. *Tree Parser*

We first compare our result to [10] as the baseline. To pursue a fair comparison, we manually divided the 18 multi-tree images and preserved the tree diagram with provided ground-truth data. Compared to the 37 samples of successful recognition reported by [10], our approach perfectly extracts 48 tree structures from the TreeRipper dataset in which 33 perfect samples are from the 79 singleton tree images (Table II).

PhyloParser is not designed to compete with perfect extraction. Our approach is developed to acquire phylogenies from big scholarly figures with high error-tolerance. We intend

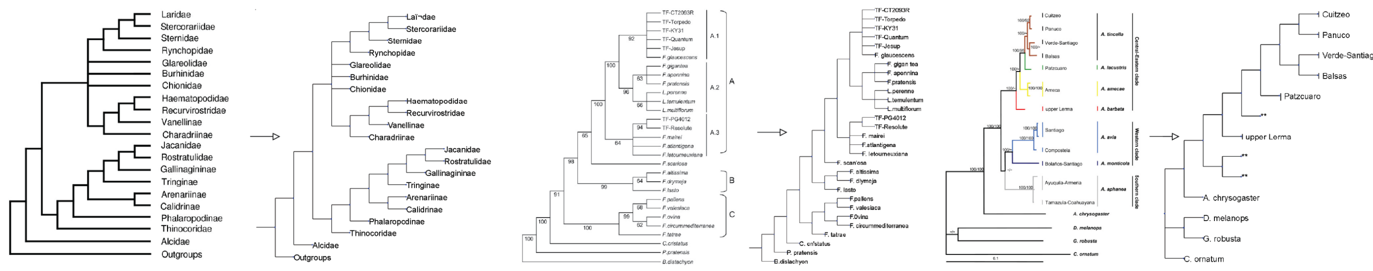

Image Source: Ericson et al., *BMC Evolutionary Biology 2003* Image Source: Hand et al., *BMC Evolutionary Biology 2010* Image Source: Ignacio et al., *BMC Evolutionary Biology 2009*

Fig. 6. Qualitative results (Left: original figure, right: regenerate figure). The left and the middle phylogenetic tree diagrams are considered perfect in our experiment without evaluating OCR results. A few species names are not converted correctly by Google Tesseract. The right sample shows a failure example that the sub-trees highlighted by light colors are missing. In the regenerate figure, we use "**" to denote a broken branch.

to correct remaining errors through an aggregate statistical analysis over a large corpus of trees as part of future work. To evaluate our approach more thoroughly, we use Zhang-Shasha distance (ZSD distance)[25] to measure the tree edit distance between the recovered tree structure and the ground-truth. ZSD distance considers three operations:

- Change one node label to another
- Delete a node. All children of the deleted node become children of its parent
- Insert a node

Different order of siblings is identified differently when measuring ZSD. It does not agree with the essential property of a phylogenetic tree, i.e. changing the order of siblings does not change the meaning of phylogeny. Nevertheless in our study, we can ignore this discrepancy because our algorithm do not switch siblings during reconstruction. We normalize the ZSD to the total number of true nodes for each phylogenetic tree, defined as an error rate. The error rate of zero indicates a perfect reconstruction. The normalized value can be larger than one when the algorithm produces many false positive branches and leaves. We do not evaluate the performance of Google Tesseract OCR engine, because it is not the main contribution of this study. Thus we name all species the same in our evaluation.

We evaluate our main tree reconstruction approach and the two methods for searching missing leaves, branches, and sub-trees. Table II shows the result obtained by using different combination of searching methods. *Base* denotes the line-based reconstruction algorithm; *A* is the method for connecting sub-trees and vertical branches; *B* is the method for associating orphan text boxes. We obtained the lowest mean error rate of 0.148 from *Base* + *A* + *B*. It can be interpreted as that approximately 15 nodes are missing, incorrectly located, or mistakenly created from a tree containing 100 nodes. Method B is a trade-off strategy because it can mistakenly create false leaves: we receive a lower error rate on average but with fewer perfectly extracted trees compared to *Base* + *A*.

Figure 5 shows the binned result in which each bin represents a group of trees corresponding to a range of error rate. The wide bars denote the counting numbers of trees. We obtained 51 (36.2%) perfect recovered trees and 108 (76.7%) trees with error rate below 0.2. The thin bars denote the average numbers of nodes for the groups of trees.The average number of node is 45 for the perfect bin (error rate = 0), which can be seen as a balanced binary tree with approximately 20 leaves. Our algorithm performs better on small trees rather than large trees. The main reason for the degradation is that components in large tree diagrams are dense or even occluded. These large and dense trees are probably a consequence of page limits for journals. The other chunks of failures are trees with edges in extremely light colors. These lines are not detected by the Hough transform and neither are the corners and joints.

Figure 6 shows the qualitative results of three phylogenetic tree diagrams. We show the original figure on the left and the regenerate figure on the right for each sample. The left and the middle samples are considered perfect in our evaluation; the right samples are partially recovered because the light lines are not successfully detected. The Google Tesseract performs well for these tree diagrams. Only few species names are not correctly converted.

V. Conclusions and Future Work

We have presented PhyloParser, a framework that automatically identifies phylogenetic tree figures from scientific literature, extracts the key components of tree structure, and reconstructs them to recover the raw data of species relationships. For the CNN-based classifier, we obtain 95% accuracy for classifying scientific figures into 8 categories and 99% precision for identifying phylogenetic tree diagrams. For the tree parsing algorithm, we obtained an average error rate of 0.15 from our testing image set containing 141 tree diagrams collected from scientific literature. Our tree parser does not handle circular trees and cladograms, but we plan to extend our algorithm to broader styles of tree diagram. To improve the OCR results, we plan to extend the Tesseract dictionary by collecting specie names from NCBI taxonomy database. We are working to extract phylogenies from big scholar data and store them in a database. We plan to fix the error nodes by comparing overlapping trees and build a scholarly search engine particular for phylogenetic research.

In future work, we aim to construct a database of species relationships automatically from the scientific literature, validate these relationships against manually constructed databases, and answer questions about the coverage and veracity of the results, and how the confidence of scientific results has changed over time.

More broadly, we aim to facilitate a variety of research projects over scientific figures, an area we call viziometrics.

It extends prior work in bibliometrics and scientometrics but focuses on the role of visual information encoding. In previous work, we developed a figure processing pipeline that automatically classifies figures into equations, diagrams, plots, photos, and tables. To facilitate further research on this visual objects, we make both the code and the data open for other researchers to explore. By integrating the figure-type labels and article metadata, we analyzed the patterns across journals, over time, and relationships to impact. In different disciplines, we found that the role of the five figure types can vary widely. For instance, clinical papers tend to have higher photo density and computational papers tend to have higher diagram and plot density. In respect to visual patterns over time, we found a growing use of plots, perhaps suggesting increasing emphasis on data-intensive methods. Our key result is that high-impact papers tend to have more diagrams per page and a higher proportion of diagrams relative to other figure types. A possible interpretation is that clarity is critical for impact: illustrating an original idea may be more influential than quantitative experimental results. We also described a new application to search and browse scientific figures, potentially enabling new kinds of search tasks. The VizioMetrics.org systems affords search by keyword as well as figure type, and shows results in a figure-centric layout. We believe more interesting and useful applications can be inspired by the concept of viziometrics. We also encourage people to use our publicly available corpus and software to explore this area of research and create a new community of interest.

Acknowledgment

This work is sponsored in part by the National Science Foundation through S2I2 award 1216879 and IIS award III-1064505, a subcontract from the Pacific Northwest National Lab, the University of Washington eScience Institute, the Metaknowledge Network funded by the John Templeton Foundation, and an award from the Gordon and Betty Moore Foundation and the Alfred P. Sloan Foundation.

References

[1] Z. Yang and B. Rannala, "Molecular phylogenetics: principles and practice," *Nature Reviews Genetics*, vol. 13, no. 5, pp. 303–314, 2012.

[2] B. T. Grenfell, O. G. Pybus, J. R. Gog, J. L. Wood, J. M. Daly, J. A. Mumford, and E. C. Holmes, "Unifying the epidemiological and evolutionary dynamics of pathogens," *science*, vol. 303, no. 5656, pp. 327–332, 2004.

[3] J. Hey, "Isolation with migration models for more than two populations," *Molecular biology and evolution*, vol. 27, no. 4, pp. 905–920, 2010.

[4] M. Kellis, N. Patterson, M. Endrizzi, B. Birren, and E. S. Lander, "Sequencing and comparison of yeast species to identify genes and regulatory elements," *Nature*, vol. 423, no. 6937, pp. 241–254, 2003.

[5] V. Morell, "Treebase: the roots of phylogeny," *Science*, vol. 273, no. 5275, p. 569, 1996.

[6] M. A. OLeary and S. Kaufman, "Morphobank: phylophenomics in the cloud," *Cladistics*, vol. 27, no. 5, pp. 529–537, 2011.

[7] A. Brady and S. Salzberg, "Phymmbl expanded: confidence scores, custom databases, parallelization and more," *Nature methods*, vol. 8, no. 5, pp. 367–367, 2011.

[8] R. P. Womack, "Research data in core journals in biology, chemistry, mathematics, and physics," *PloS one*, vol. 10, no. 12, p. e0143460, 2015.

[9] T. Laubach and A. Von Haeseler, "Treesnatcher: coding trees from images," *Bioinformatics*, vol. 23, no. 24, pp. 3384–3385, 2007.

[10] J. Hughes, "Treeripper web application: towards a fully automated optical tree recognition software," *BMC bioinformatics*, vol. 12, no. 1, p. 178, 2011.

[11] X. Lu, J. Wang, P. Mitra, and C. L. Giles, "Automatic extraction of data from 2-d plots in documents," in *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, vol. 1. IEEE, 2007, pp. 188–192.

[12] S. Kataria, W. Browuer, P. Mitra, and C. L. Giles, "Automatic extraction of data points and text blocks from 2-dimensional plots in digital documents." in *AAAI*, vol. 8, 2008, pp. 1169–1174.

[13] M. Savva, N. Kong, A. Chhajta, L. Fei-Fei, M. Agrawala, and J. Heer, "ReVision: Automated Classification, Analysis and Redesign of Chart Images," in *UIST '11*, 2011, pp. 393–402.

[14] N. Siegel, Z. Horvitz, R. Levin, S. Divvala, and A. Farhadi, "Figureseer: Parsing result-figures in research papers," in *European Conference on Computer Vision*. Springer, 2016, pp. 664–680.

[15] A. Kembhavi, M. Salvato, E. Kolve, M. Seo, H. Hajishirzi, and A. Farhadi, "A diagram is worth a dozen images," in *European Conference on Computer Vision*. Springer, 2016, pp. 235–251.

[16] A. Rambaut, "Treethief: a tool for manual phylogenetic tree entry," *Program distributed by the author. http://evolve. zoo. ox. ac. uk/software/TreeThief/main. html*, 2000.

[17] T. Laubach, A. von Haeseler, and M. J. Lercher, "Treesnatcher plus: capturing phylogenetic trees from images," *BMC bioinformatics*, vol. 13, no. 1, p. 110, 2012.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[20] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 675–678.

[21] J. Matas, C. Galambos, and J. Kittler, "Robust detection of lines using the progressive probabilistic hough transform," *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 119–137, 2000.

[22] R. Smith, "An overview of the tesseract ocr engine," in *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, vol. 2. IEEE, 2007, pp. 629–633.

[23] P.-s. Lee, J. D. West, and B. Howe, "Viziometrics: Analyzing visual information in the scientific literature," *arXiv preprint arXiv:1605.04951*, 2016.

[24] ——, "Viziometrix: A platform for analyzing the visual information in big scholarly data," in *Proceedings of the 25th International Conference Companion on World Wide Web*. International World Wide Web Conferences Steering Committee, 2016, pp. 413–418.

[25] K. Zhang and D. Shasha, "Simple fast algorithms for the editing distance between trees and related problems," *SIAM journal on computing*, vol. 18, no. 6, pp. 1245–1262, 1989.