2017 14th IAPR International Conference on Document Analysis and Recognition

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, billhowe}@uw.edu

***Abstract*—We consider a new approach to extracting information from dendrograms 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*—*Dendrogram, 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 a larger research agenda, we are working to develop a general framework for information extraction from these visual elements in the 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 8). 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 about 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 than 40 thousands phylogenetic trees available on PubMed Central, 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.

Our approach focuses on rectilinear dendrograms, which account for 75% of all phylogenetic trees in a representative sample of the scientific literature. We automatically recognize the fundamental components: horizontal branches, vertical branches and the text of species names. We use Hough transforms 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 top-down and bottom-up. This approach allows us to ignore noise at the 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

2379-2140/17 $31.00 © 2017 IEEE
DOI 10.1109/ICDAR.2017.180

error appear insurmountable.

In this paper, we make the following contributions.

- We present a hybrid algorithm for parsing dendrograms that recognizes the components of the diagram then reassembles them into a complete tree. This topological approach is designed to be more resistant to noise effects that limit the effectiveness of previous approaches.
- We evaluate this algorithm on real datasets used in prior work as well as a new dataset extracted from one million papers in PubMed. We find that our approach produces more perfect reconstructions than the state-of-the-art, and can also perform well on a significantly larger and more diverse dataset than has been previously tested.
- While previous methods used datasets assembled by hand, we train and evaluate two deep neural network architectures (ResNet-50 [11] and AlexNet [12]) to automatically recognize phylogenetic tree diagrams in the scientific literature.
- We organize all of these methods into an end-to-end system called PhyloParser that automatically extracts phylogenetic relationships in a machine-readable format from any set of images.
- We released PhyloParser and all datasets we use in our experiments online under permissive licenses 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

A number of studies have investigated techniques for extracting information from figures and tables. Early work from Futrelle et al. [13], Zhou et al. [14], and Huang et al. [15] focus on chart classification, but more recent work emphasizes the extraction of quantitative data from scientific charts, including 2-D line charts [16], [17], [18], [19], bar charts [20], [21], [22], and tables [23]. Elzer et al. studied the intended message of bar and line charts [24], [25]. In 2011, Savva et al. proposed ReVision, a system that automatically redesigns bar charts and pie charts to improve graphical perception [20]. More recently, Chen et al. proposed DiagramFlyer [26], which facilitates search over scientific figures given a x-label, y-label, and corresponding ranges. In 2017, we proposed Viziometrics [27], a system for mining figures at scale in order to study the relation between graphical communication and scientific impact.

These projects focused on parsing plots that present quantitative data, but parsing diagrams is increasingly recognized to be even more challenging. In 2016 Kembhavi et al. proposed Diagram Parse Graphs, a model for parsing and studying semantic interpretation of diagrams that illustrates relationships between nature objects [28]. Compared to natural images, visual illustrations encode rich knowledge that has been well organized, which are potentially sources for artificial intelligence studies.

In 2000, Rambaut proposed the first interactive tool to convert a tree image into machine-readable format [29]. 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 [30]. 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.

Since we first submitted this paper, Murray-Rust et al. proposed a method for parsing 4336 rectilinear dendrograms manually collected from the International Journal of Systematic and Evolutionary Microbiology (IJSEM) [31]. The authors state that they selected dendgrograms from this journal because the the diagrams in IJSEM are consistent and typically created from the same software. Further, the parsing algorithm is not explained in detail, as the focus of their work was on constructing a shared database.

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 the following prerequisites, which are commonly 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 than 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

Seigel et al. [19] tuned a pre-trained AlexNet and ResNet-50 using 60,000 scientific figures and showed that using deep neural networks significantly improves the classification of scientific figures compared to the previous state-of-the-art approach [20]. Their model focuses on classifying plot graphs such as bar plots, line plots, etc. We train the neural networks on our own dataset to classify images into one of eight categories (equations, photos, diagrams, tables, plots, electrophoresis gels, metabolic pathways and phylogenetic trees), but in this paper, we are only interested in the figures labeled as phylogenetic trees. We describe the training dataset in more detail in Section IV-A.

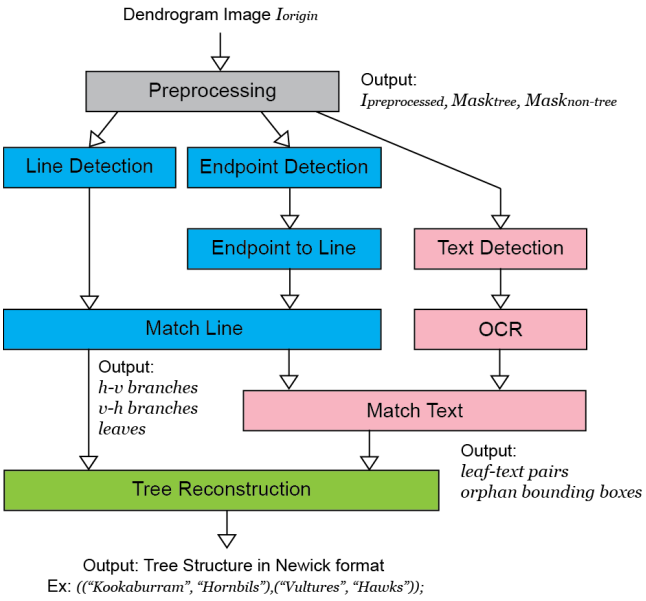

Fig. 1. Pipeline of parsing a tree diagram. There are four stages: preprocssing, tree component extraction, text extraction, and tree reconstruction. First, we remove color background and separate text from tree diagram (gray box). Next, we extract tree components (blue boxes) and specie names (pink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

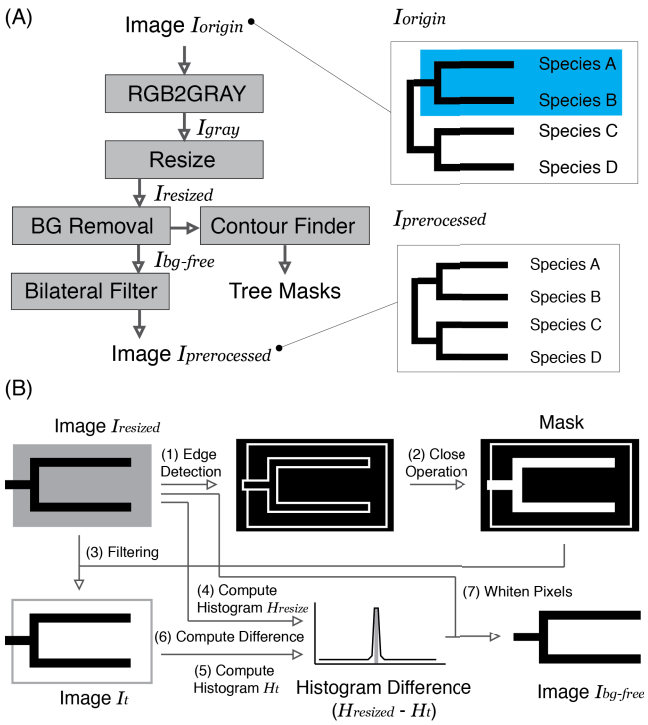

Fig. 2. (A) Preprocessing steps that take the original image as input and produce an image in gray-scale with any background removed, as well as two masks: one for the tree pixels and one for the non-tree pixels. (B) To remove the background patches, we use Canny edge detection followed by a morphological close operation to produce a mask that covers the tree structure as well as any text. We then apply the mask on the resized image $I_{resized}$ to produce a temporary output I_t. Next we compared the histogram difference between I_t and $I_{resized}$ to determine the intensity ranges of the color patches. Finally we remove the background patches in $I_{resized}$ by whitening the pixels in these intensity ranges.

We train the networks using the Caffe framework [32] installed on an Amazon EC2 instance (g2.2xlarge). Iteration stops when the accuracy of testing on validation data is steady and invariant to decreasing learning rate. We describe the experimental results in Section IV-B.

B. Parsing Phylogenetic Trees

Figure 1 illustrates the pipeline of our parsing algorithm. We divide the algorithm into four stages: preprocessing (gray), 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 to minimize parsing errors overall; our preprocessing steps are therefore designed to optimize line detection.

Figure 2(A) gives an overview of the preprocessing steps. The original image I_{origin} is converted into the gray-scale (I_{gray}) in the beginning. To make our algorithm scale invariant, we resize I_{gray} 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 opening 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 I_{gray} (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. This operation produces $I_{resized}$ as the output. Although rare, the method will remove extra features if colors of the tree topology and the background in the original image are highly similar.

Next, we remove color patches in the background of $I_{resized}$. This process is illustrated in Figure 2(B). The color patches decrease the gradient of tree boundaries, making it difficult for the line detector to identify segments reliably. To detect color patches, we use the Canny edge detection to extract all boundaries, followed by a morphological close operation with a 5×5 rectangular kernel to backfill the elements.

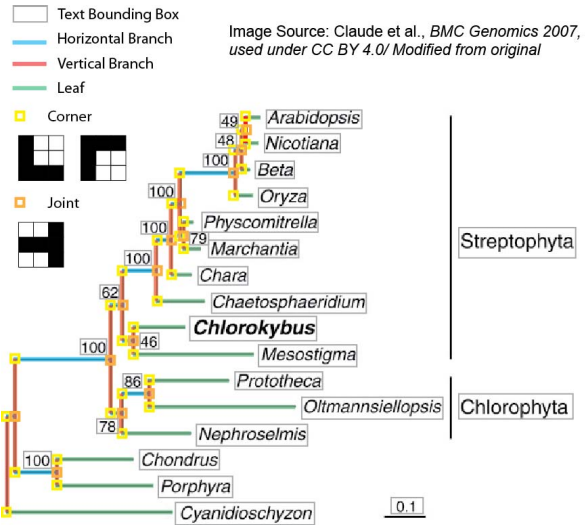

Fig. 3. Basic components of a dendrogram. We use the Hough transform to detect vertical lines and horizontal lines. To capture short lines that are likely not captured by the Hough transform, we further detect joints and corners using convolution operation and then match them to obtain more lines.

We use the output of this operation as a binary mask and apply it on $I_{resized}$ to produce a temporary image I_t. The color patches in I_t are partially removed, so we can obtain the intensities of color patches by comparing $I_{resized}$ and I_t. We compute the difference between the intensity histograms of I_t and $I_{resized}$ and extract the ranges of intensities around the peaks using the peakutils package from OpenCV. Finally, we whiten the pixels in $I_{resized}$ with the intensity within these ranges. This operation produces $I_{bg\text{-}free}$ as the output.

A common source of errors for line detection is text, so we need to separate text from the tree structure. We acquire contours in $I_{bg\text{-}free}$ by using the contour finder in OpenCV, then use the heuristic that the tree structure is typically the largest contour in an image. This largest contour is used to build a *tree mask*, and the remaining contours are used to build a *non-tree* mask. We apply the two masks to distinguish the main structure of the tree from other sources of lines, including text.

Finally, we apply a bilateral filter on $I_{bg\text{-}free}$ to sharpen edges, which improves line detection specifically for thin lines. The final output $I_{preprocessed}$ of the preprocessing will be used as the input of the next stages.

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 regions in $I_{preprocessed}$ and binarize the filtered image with a threshold of 30. We detect lines in the binary image using two approaches: Hough transform [33] and endpoint detection.

The Hough transform can detect lines at given angles. 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:

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

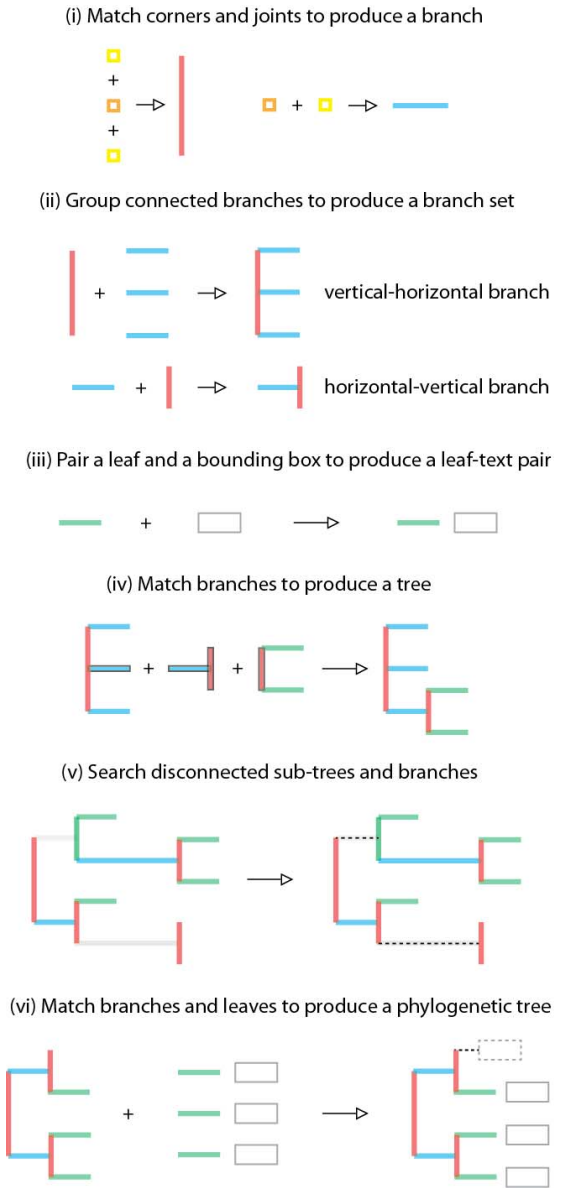

Fig. 4. Reconstruction logic. Our parsing algorithm is based on accurate line detection. Beside using the Hough transform to detect lines, we also extract lines by detecting corners and joints as shown in part (i) of (B). Texts are located by contour finder and converted by Google Tesseract. In (B) We visualize the concept of "Match Line" in part (ii), "Match Text" in part (iii) and "Tree Reconstruction" in part (iv) to (vi).

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

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

A Hough-transform-based line detection, such as the one

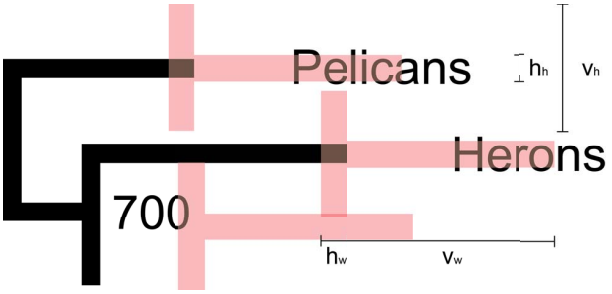

Fig. 5. To identify leaves, we train a model based on spatial features between lines and text. We design the features to capture the unique pattern of leaves: a vertex followed by text. We extract the raw pixels (colored red), along with the horizontal distance between the right endpoint of a leaf and 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 $v_h = 9$.

we use here, does not guarantee 100% recall. To increase the recall, we capture the corners and joints (see Figure 3) 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 3 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 4(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 3 shows the ideal result of line detection and corner detection. Next, we connect vertical lines and horizontal lines to create branch sets (Figure 4(B)(ii)). We associate each vertical line 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 5. 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 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 a random forest that gives 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: The contours in the non-tree mask represents the locations of characters, strings, and any other irrelevant items such as arrows, photos, etc. For each contour,

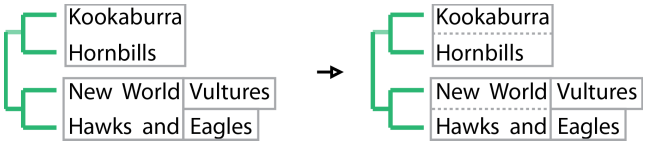

Fig. 6. Method of separating cross-line bonding box. Text in different lines can be bonded together if they are very close. A reasonable segmentation can be deducted from (1) other bonding boxes or (2) corresponding leaves. We seek the first solution prior to the second solution, because the perfect segmentation from the second solution only applicable when all corresponding leaves are found.

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 4(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 6). 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 6), 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 that approximately 50% of our test images need such process to separate at least one bounding box.

Some text may overlap the tree structure, and therefore be erroneously considered part of the tree mask and be ignored by the OCR step. To recover this text, we use the fact that we know where the leaf ends after extracting horizontal lines. Starting from this point, we scan the area to the right with a 10-pixel-high box to seek missing text. Any recovered text is then associated with the leaves and is used as the species names in the next stage, while any text not associated 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 v-h-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 4(B)(v) shows an example in which three sub-trees are not linkable because of undetected lines (highlighted in light gray). We reconnect the upper sub-tree and the lower

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** [20]
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%

vertical branch using method A. Method B handles a particular tree style that a tree does not use horizontal lines to tip species, for instance the lower part in Figure 4(B)(v) and the top species in Figure 4(B)(vi). For a broken branch, we associate it with the orphan text boxes within a distance of 15 pixels, defined by observation. Method B 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 only need to traverse the tree structure to produce the final tree string in the Newick format.

IV. EVALUATION

A. Image Dataset

We use the annotated data from our previous work [27] and additionally labeled 1411 electrophoresis gels, 1119 metabolic pathways and 1308 phylogenetic trees to create a training corpus. We finally compiled an image set containing 18778 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 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[1]. 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.

[1]Circular trees and cladograms approximately accounts for 21.5% and 3.5% phylogenetic trees respectively in the scientific literature.

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 / 141
Base + A	0.156	**54** / 141
Base + A + B	**0.148**	51 / 141
TreeRipper Dataset [10]		
TreeRipper	N/A	37 / 114
Base + A	**0.116**	**48** / 100
Base + A + B	0.120	44 / 100

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 [12]. 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.

Both of these network architectures delivered higher accuracy than previous state-of-the-art approaches [20]. In contrast to Siegel et al [19], we obtained the highest accuracy from AlexNet trained from scratch. In general, classifying diagrams and the corresponding sub-types are more difficult than other types of figures due to the higher diversity, showing in all the three 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 the work of Hughes [10] as the baseline. The author released a set of 100 images with the ground-truth parsed data.[2] In addition, there are 21 images containing two or more dendrograms respectively. We manually segmented the 21 compound images and preserved the one with provided ground-truth data. The author reported 37 trees that are successfully recognized. However, the author does not precisely define the meaning of "successfully recognized." We consider the definition of the successful case as "every node of the tree is correctly extracted" i.e. "perfect extraction" according to the output of TreeRipper. Our approach achieves perfect extraction from 48 dendrograms in the published TreeRipper dataset (Table II).

Since our ultimate goal is to build a database of phylogenetic relationships, we can tolerate partial extractions in which not every node in a dendrogram is successfully recognized.

[2]This dataset does not appear to be the exact one used in the authors' paper, as that dataset contained 114 images.

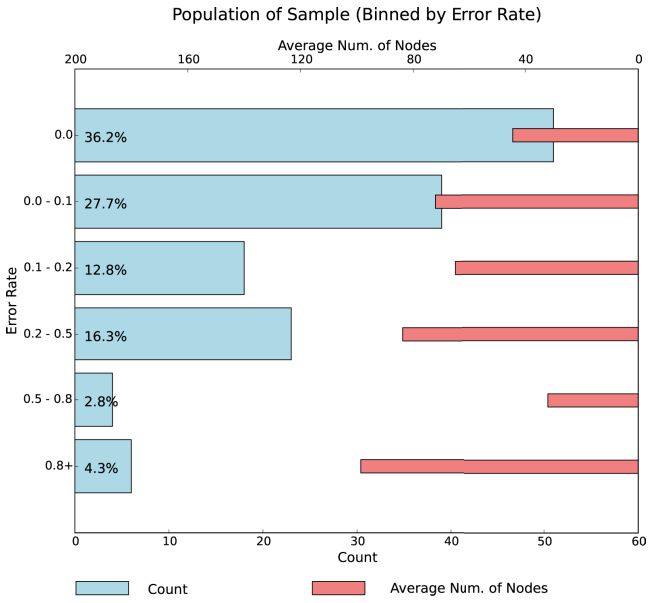

Fig. 7. Histogram of reconstructed trees categorized by error rates. The wide bars show the figure counts and the thin bars show the average number of nodes. We obtain 51 perfectly reconstructed trees and 108 reconstructed trees with error rates below 0.2. The performance of our algorithm degrades when the size of tree increases.

That is, any relationships we can extract from a dendogram is potentially useful. And, we can potentially recover individual misparsed errors using an aggregate statistical analysis over a large corpus of trees, but this step is out of scope for the current paper. For these reasons, we need a metric other than the percentage of perfect extractions if we are to assess the performance of our approach. We therefore use the Zhang-Shasha distance (ZSD distance) [34] 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

One problem with this approach is that ZSD is sensitive to the order of the siblings, but phylogenetic trees are not: changing the order of siblings does not change the phylogeny. We can ignore this issue, however, because our algorithm naturally preserves sibling order during reconstruction.

We normalize the ZSD to the total number of true nodes for each phylogenetic tree, defined as an error rate. An error rate of zero indicates a perfect reconstruction. The normalized value can be greater than 1.0 when the algorithm produces many false positive branches and leaves. We do not evaluate the performance of the Google Tesseract OCR engine, because it is not the main contribution of this study. We therefore refer to all species with a single name 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. Since the released TreeRipper is not functional, we cannot evaluate the error rate of TreeRipper. *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 7 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 likely erased by binarization or not detected by the Hough transform and neither are the corners and joints.

Figure 8 shows the qualitative results of three phylogenetic tree diagrams. We show the original figure on the left and the regenerated figure from the Newick output 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 a 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 will correct the spelling errors by comparing the string to existing taxonomy databases and the full-text content of the original papers via minimum edit distance. We are working to extract phylogenies from big scholar data. The structure errors can be determined when we merge the thousands of the trees; the supertree will therefore be probabilistic in consideration of structure errors. 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.

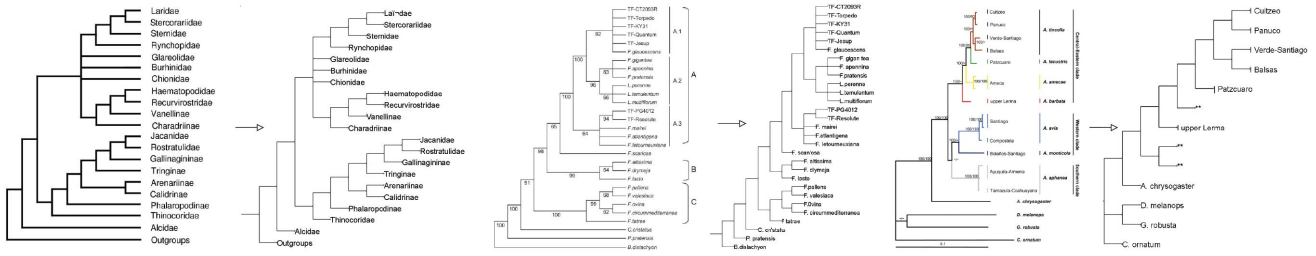

Fig. 8. Left: original figure, right: Re-visualization of the extracted phylogenies (the actual results are produced in the machine-readable Newick format). The left and the middle phylogenetic tree diagrams are considered perfectly recovered in our experiment without evaluating OCR results. A few species 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.

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.

[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] 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.

[12] 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.

[13] R. Futrelle, M. Shao, C. Cieslik, and A. Grimes, "Extraction,layout analysis and classification of diagrams in pdf documents," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2003.

[14] M. Shao and R. P. Futrelle, "Recognition and classification of figures in pdf documents," in *International Workshop on Graphics Recognition*. Springer, 2005, pp. 231–242.

[15] W. Huang and C. L. Tan, "A system for understanding imaged infographics and its applications," in *Proceedings of the 2007 ACM symposium on Document engineering*. ACM, 2007, pp. 9–18.

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

[17] 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.

[18] S. R. Choudhury and C. L. Giles, "An architecture for information extraction from figures in digital libraries." in *WWW (Companion Volume)*, 2015, pp. 667–672.

[19] 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.

[20] 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.

[21] R. A. Al-Zaidy and C. L. Giles, "Automatic extraction of data from bar charts," in *Proceedings of the 8th International Conference on Knowledge Capture*. ACM, 2015, p. 30.

[22] R. A. Al-Zaidy and L. C. Giles, "A machine learning approach for semantic structuring of scientific charts in scholarly documents," in *Twenty-Ninth IAAI Conference*, 2017.

[23] J. Fang, P. Mitra, Z. Tang, and C. L. Giles, "Table header detection and classification." in *AAAI*, 2012, pp. 599–605.

[24] S. Elzer, S. Carberry, I. Zukerman, D. Chester, N. Green, and S. Demir, "A probabilistic framework for recognizing intention in information graphics," in *International Joint Conference On Artificial Intelligence*, vol. 19. LAWRENCE ERLBAUM ASSOCIATES LTD, 2005, p. 1042.

[25] S. Elzer, S. Carberry, and I. Zukerman, "The automated understanding of simple bar charts," *Artificial Intelligence*, vol. 175, no. 2, pp. 526–555, 2011.

[26] Z. Chen, M. Cafarella, and E. Adar, "Diagramflyer: A search engine for data-driven diagrams," in *Proceedings of the 24th International Conference on World Wide Web*. ACM, 2015, pp. 183–186.

[27] P. Lee, J. D. West, and B. Howe, "Viziometrics: Analyzing visual information in the scientific literature," *IEEE Transactions on Big Data*, 2017.

[28] 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.

[29] 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.

[30] 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.

[31] R. Mounce, P. Murray-Rust, and M. Wills, "A machine-compiled microbial supertree from figure-mining thousands of papers," *Research Ideas and Outcomes*, vol. 3, 2017.

[32] 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.

[33] 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.

[34] 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.