An Architecture for Information Extraction from Figures in Digital Libraries

Sagnik Ray Choudhury
Information Sciences and Technology
Pennsylvania State University
sagnik@psu.edu

C Lee Giles
Information Sciences and Technology
Pennsylvania State University
giles@ist.psu.edu

ABSTRACT

Scholarly documents contain multiple figures representing experimental findings. These figures are generated from data which is not reported anywhere else in the paper. We propose a modular architecture for analyzing such figures. Our architecture consists of the following modules: 1. An extractor for figures and associated metadata (figure captions and mentions) from PDF documents; 2. A Search engine on the extracted figures and metadata; 3. An image processing module for automated data extraction from the figures and 4. A natural language processing module to understand the semantics of the figure. We discuss the challenges in each step, report an extractor algorithm to extract vector graphics from scholarly documents and a classification algorithm for figures. Our extractor algorithm improves the state of the art by more than 10% and the classification process is very scalable, yet achieves 85% accuracy. We also describe a semi-automatic system for data extraction from figures which is integrated with our search engine to improve user experience.

1. INTRODUCTION

Most scholarly documents contain multiple figures such as line graphs, scatter plots, bar graphs etc. Even though they are rich resources of information, they have not received much attention yet[3]. We propose a complete architecture for analyzing these figures. Specifically, we are interested in following problems:

1. Can we extract figures and associated metadata (figure captions, mentions) from documents accurately?

2. Can we use the extracted figures and metadata to build a better scholarly information retrieval system?

3. Can we process the figures to understand their types? What are the specific challenges in automated data extraction from such figures?

4. Given the figure data and metadata, can we understand the intended message of the figure?

The architecture is shown in figure 1. The input to the system is a born digital (non-scanned) PDF document. The document analysis module extracts the figures and associated metadata from the input PDF (see section 3). Output of the document analysis module is used by two other modules: 1. Search engine module (section 5) and 2. Image processing module (section 4).

Our search engine module is built on the captions and mentions extracted from chemistry journal articles and has been reported earlier in[6]. Here, we describe a web-based, semi-automatic system for data extraction from figures, which is integrated with the search engine to improve the user experience.

Input to the image processing module is a [figure,metadata] tuple, and output is a [figure-data, metadata] tuple. Many figures in scholarly documents represent experimental data that cannot be found in the text of the document. Automatic extraction of that data can be of huge significance. Naturally, it is a hard problem. But, more importantly, a fully automatic system should first identify the figure type (i.e., whether the figure is a scatter plot or line graph) because the data extraction algorithm would depend on that. Also, in some cases, segmentation of images before classification might be necessary.

Currently, we are only interested in a binary classification problem, where we classify each image as of a line graph or not. Our interest in line graphs stems from two facts: 1. Line graphs are abundant in scholarly papers, and 2. They mostly represent experimental data. We describe our work on segmentation and classification of figures in section 4. Our current research focus is improving the image processing module to perform fully automatic data extraction from line graphs and we discuss the challenges here. In the future, we can incorporate other types of figures (bar graphs and scatter plots) in the architecture.

In the natural language processing module we plan to generate a machine readable representation of scholarly figures. Previously, researchers in the linguistic community have analyzed bar charts and line graphs from the Web to understand their intended meaning[4]. For example, most line graphs containing multiple curves intend to show that certain experimental methods (algorithms, technique) perform better than the other baselines. To understand that semantics, we need to understand the data from which the figure was generated. Therefore, the natural language processing module in our architecture is designed to use the data ex-

WWW 2015 Companion, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3473-0/15/05.
http://dx.doi.org/10.1145/2740908.2741712 .

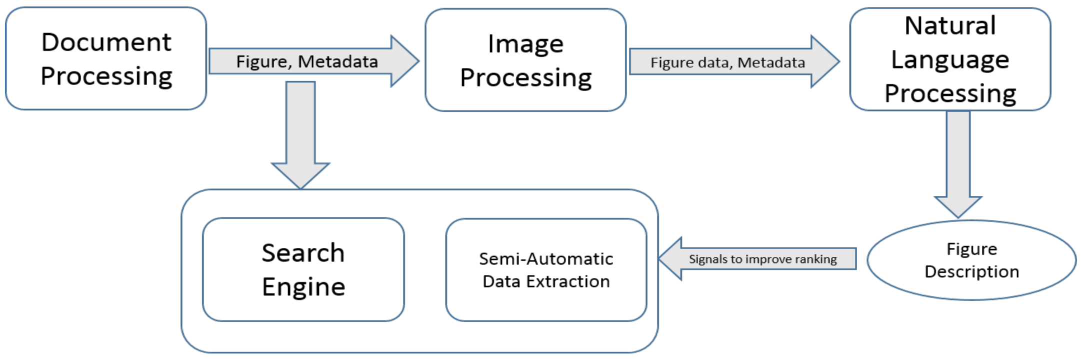


Figure 1: Architecture of our system.

tracted by the previous modules. Apart from the extracted data, we need to analyze the caption and mention to identify the entities, their coreference and relations. Most previous works in this domain have analyzed figures collected from the Web. Those figures can be analyzed without domain knowledge, whereas, scholarly figures require domain expertise. This makes our problem considerably harder than previous works.

2. RELATED WORK

Figures are embedded in PDF documents in raster (PNG, JPEG) or vector formats (SVG, EPS). Typically, raster images are embedded in PDF as separate content streams (XObjects). Therefore, it is easy for a PDF parser to extract raster images. However, it is hard to extract vector graphics. In a PDF document, graphics and textual elements are often interleaved in the content stream. An image written in vector format can also contain raster graphics elements. Researchers have approached this problem in two ways: 1. Graphics paths in a PDF are combined heuristically to produce image regions, and 2. PDF pages are converted into images and segmented into text/graphics regions using segmentation algorithms. Raster and vector graphics are treated differently in the first approach, but not in the second approach. This approach was more suitable for us as it did not involve heuristics; also, the bitmap and vector graphics in PDFs could be extracted uniformly as bitmap images, removing the need for a post processing step involving vector to raster conversion.

Document page image segmentation is an active research area since 1980[10] with several approaches proposed till date. Among them, the multi-resolution morphology based approach has several benefits[2] and is implemented in a popular image processing software called Leptonica [1]. In this approach, a halftone mask is created through a set of morphological operations, and this mask is applied on the original image to extract graphics regions. Chao et al.'s work did not specifically mention which segmentation algorithm they use, therefore, we used Leptonica for our text/graphics region segmentation. Our contribution is to improve the segmentation accuracy of Leptonica using machine learning techniques.

There has been previous work on classifying figures. Some previous works focused on 2D/non 2D plot binary classification[1] and some addressed multi class classification (line graph, bar graph, scatter plot etc)[12]. A recent work by Savva et al.[12] showed that unsupervised feature learning can produce better results than complicated feature engineering[11]. Our classifier is even simpler and scalable yet achieves 85% accuracy on a binary classification task.

[1]http://www.leptonica.com/

3. DOCUMENT ANALYSIS

This section describes our document analysis module. Input to this module is a born digital PDF document. This PDF document is analyzed to extract figures and associated metadata (figure caption, mention).

3.1 Extraction of Figures: Generic Approach

We follow Chao et al.'s approach[5] for extraction of figures from PDF documents, where document pages are converted into images and processed through an image segmentation algorithm to identify text and graphics regions. We improve previous work by two important modifications: 1. Pre-processing of the document and 2. Grouping the output of the image segmentation algorithm. Our approach is explained below.

1. **Removal of text**: In PDF documents, text is usually written using the text painting operator "Tj;" therefore, it can be easily identified in PDF content stream. As a preprocessing step, our system removes all text characters from the input PDF document.
2. **Conversion of document to page images**: Using ImageMagick[2], our system converts each page in the PDF document to images.
3. **Using image segmentation to find bounding boxes of graphics regions**: Each page image obtained in the previous step is processed through Leptonica[3] to obtain graphics regions.
4. **Improving image segmentation accuracy**: The output of Leptonica is analyzed to examine whether or not the segmentation accuracy could be improved.

Though we use Leptonica, other page segmentation approaches, such as recursive X-Y cut[10], can be used within our framework. Most page segmentation algorithms fail when the text and image regions do not have significantly different pixel densities[2]. Fortunately, our pre-processing step solves this problem.

[2]http://www.imagemagick.org/
[3]http://www.leptonica.com/

3.2 Improving Text/Graphics Segmentation

3.2.1 *Problem Description and Dataset*

When page images are processed through Leptonica, a set of bounding boxes are produced. If the output is only one bounding box, there is no need to process the output further. When there are multiple bounding boxes, there can be two cases:

- Each bounding box corresponds to a unique graphics region. This can happen when there are multiple figures on the page, and Leptonica correctly identifies each of them separately. See figure 2a for an example. We do not need to process these outputs further.
- For a single graphics region, multiple bounding boxes are generated by Leptonica. See figure 2b for an example. In this case, the output needs to processed further to produce actual graphics regions.

In the second case, actual number of graphics regions in a page need to be determined before combining the bounding boxes. Fortunately, most figures in scholarly documents contain a caption. Number of graphics regions (which is same as number of figures) can be determined using simple regular expressions.

To test our hypothesis, we randomly sampled 100 born digital PDF documents from CiteSeerX repository[4], and removed text from them. We split these pre-processed documents into pages and converted them into images, yielding approximately 1800 page images. We randomly sampled 300 page images from this dataset.

3.2.2 *A K-means Based Approach*

We used a simple variant of K-means technique to group the bounding boxes. For each page image, we had N 4-dimensional data points (co-ordinates of the bounding boxes), which had to be grouped in K clusters (K $\leq$ N). K-means aims to partition N observations into K sets (K $\leq$ N) to minimize the within-cluster distance:
$\underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} Distance(\mathbf{x}_j - \boldsymbol{\mu}_i)$
where μ_i is the mean of points in S_i, and $Distance(\mathbf{x}_j - \boldsymbol{\mu}_i)$ measures the distance between data points and mean of the cluster.

It is easy to see that N data points can be divided into K exclusive partitions in $\binom{N-1}{K-1}$ ways. For large N and K, it is computationally infeasible to enumerate all possible combinations. Therefore, implementations of K-means technique use Lloyd's algorithm, which is a greedy way to reach a local minimum. Apart from the local minimum problem, Lloyd's algorithm suffers from two important drawbacks: 1. It is heavily dependent on the choice of initial K data points and 2. The algorithm is guaranteed to converge only when the distance function is Euclidean distance, i.e. $Distance(\mathbf{x}_j - \boldsymbol{\mu}_i) = \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2$.

However, a complete enumeration is guaranteed to reach the global minimum and should not depend on any other parameter. In our case, both N and K would be very small (largest N=50 and largest K=6 in our dataset); so, enumerating all possible cases was quite feasible. Also, this gave us opportunity to experiment with multiple distance functions. For a particular configuration (a partition of N points in K clusters), distance-value(configuration) is the sum of intra-cluster distances. We experimented with six distance functions: 1. **Euclidean-center**: Sum of Euclidean distance between central points of bounding boxes in a cluster and the cluster center. 2. **Euclidean-pairwise**: Sum of pairwise Euclidean distance between central points of bounding boxes in a cluster center. 3. **Manhattan-center**: Sum of Manhattan (taxicab) distance between central points of bounding boxes in a cluster and the cluster center. 4. **Manhattan-pairwise**: Sum of pairwise Manhattan distance between central points of bounding boxes in a cluster center. 5. **Rectangular-Manhattan-center**: Sum of Manhattan distance between bounding boxes in a cluster and the cluster center. and 6. **Rectangular-Manhattan-pairwise**: Sum of pairwise Manhattan distance between bounding boxes in a cluster center.

We evaluated the clustering quality by average adjusted rand index (ARI) value. Table 1 shows the ARI values for different distance functions. An ARI value of 1 denotes a perfect clustering implying all figures in that page have been extracted perfectly. Euclidean-center, i.e., sum of euclidean distance between central points of bounding boxes in a cluster and the cluster center clearly outperforms others.

Distance function	**Average ARI value**	**Number of perfect clusterings**
Euclidean-center	**0.80**	**29**
Euclidean-pairwise	0.63	19
Manhattan-center	0.63	19
Manhattan-pairwise	0.63	21
Rectangular-Manhattan-center	0.52	15
Rectangular-Manhattan-pairwise	0.52	15

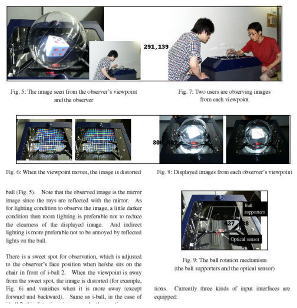
Table 1: Results for clustering of bounding boxes for different distance functions.

3.2.3 *Results and Discussions*

Among our 300 page images, 230 contained figures and captions. Following Chao et al.[5]'s approach, we got 110 correct results. Among the 120 errors (cases where the page contained a single figure and the output contained multiple boxes), 71 were corrected just by analyzing the output files. This amounts to an improvement of nearly 32% over the earlier work. The other 49 page images contained multiple figures, and in 42 of them, there was at least one figure for which multiple bounding boxes were produced. A further improvement of nearly 10% was achieved by our clustering algorithm (in the 29 cases where the clustering was perfect).

We measure the segmentation accuracy in the page level, i.e. a segmentation is considered a failure when all figures in the page are not identified correctly. Therefore, our results are conservative. Another common measure for segmentation accuracy is pixel-level accuracy[2], where each pixel is classified as either an element of a graphics region or not. This measure suffers from the problem that the boundary

[4]`http://citeseerx.ist.psu.edu`



(a) An example where the image segmentation algorithm produces graphics regions correctly.

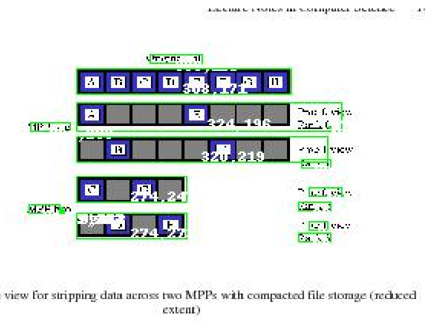


Fig 6. File view for stripping data across two MPPs with compacted file storage (reduced extent)

(b) An example where the image segmentation algorithm does not produce graphics regions correctly. Each green region is an output produced by Leptonica. The bounding boxes need to be grouped to produce the final graphics region.

Figure 2: Example outputs from Leptonica.

pixels of graphic regions are given the same importance as the inside pixels. We are in the process of developing a better accuracy measure and gold standard data.

3.3 Extraction of Figure Metadata and Matching

Once the figure and its location in the page are extracted, we are left with two tasks: 1. Figure metadata (caption and mention) extraction, and 2. Matching figures with extracted metadata. In our previous works[6], we have described two approaches for these problems. It is easy to find locations of a text line in a PDF page. Therefore, matching a figure with a caption is not very hard. We observed, in most cases, the caption followed the figure.

4. IMAGE PROCESSING MODULE

Input to this module is a [image,metadata] tuple that can be analyzed to extract data from the figure. For example, a curve in a line graph is generated from a set of data points (x,f(x)). Reverse engineering the process is naturally hard. But, before the data extraction, two important sub-problems need to solved. The image might contain multiple figures that have to be separated. We use recursive X-Y cut algorithm[10] to retrieve these sub-figures. Horizontal and vertical profiles of a binarized image is created by summing up the foreground pixels in X and Y direction. Then the image is segmented in the first "valley" location where the profile has a zero value. This process is repeated until there is no such valley. For each sub-figure, we also need to understand its type and in the next section 4.1 we propose a scalable algorithm for that.

4.1 Image Classification

As we are interested in data extraction from line graphs, we developed a binary classifier to determine whether or not an image is a line graph. Though there has been considerable work on this problem, most of the earlier methods involve complicated feature engineering. Recent progress in unsupervised feature learning motivates simpler feature designs.

Dataset: We randomly selected 478 images from 90,000 images extracted by our extractor and manually labeled them as a line graph or not. We divided the whole dataset into training (424 images) and test (54 images). We tuned the parameters of our model on the training dataset through 5-fold cross validation, and finally trained the model on the whole training data.

4.1.1 *Feature Extraction*

We followed the feature extraction process from Coates et al.[7] and Savva et al.[12] with necessary modifications. First, we scaled each image in our dataset to a dimension of 128x128 and converted them into binary images. We extracted patches of size 4x4 (a contiguous set of pixels) from each image. Our goal was to keep the patch dimension as small as possible, and, at the same time, keep the total number of patches reasonable. From an image of dimension 128x128, there are up to 1024 patches of 4x4 sizes, which was a reasonable number for our next step. From the 1024 patches extracted in the previous step, we selected N patches randomly. After experimenting with N=50,100,150 and 200, the value of N was determined to be 100 (see section 4.1.2). We represented each patch as a 16 (4x4) dimensional feature vector and performed K-means clustering to get K cluster centers. After experimenting with K=5, 10 and 15, the value of K was determined to be 5. After the feature extraction step, each image was represented as an 80 dimensional feature vector by concatenating five cluster centers of dimension 16 together.

4.1.2 *Experiments and Results*

We used a linear kernel SVM as our classifier to determine the value of N (number of patches) and K (number of cluster centers). We experimented with four popular discriminative models: LDA, QDA, linear kernel SVM and Random forest with 100 decision trees. From fivefold cross validation results on our training data, it was clear that SVM outperformed other classifiers in almost all evaluation metric, and QDA was not suitable at all. Therefore, in further experiments, we considered the linear kernel SVM as our classifier. Neither using SVM with other kernels (such as rbf), nor increasing number of decision trees in random forest, did improve classification accuracy. Further, using PCA and cross-validation, we discovered that ten was the optimal number of dimension for our dataset. Finally, the linear kernel SVM with ten principal components was trained on the entire training data, and the evaluation results for test data are reported in 2. As can be seen, the test accuracy is 0.85, which is comparable to a similar work by Browuer et al[1].

		Actual Classes		
		Positive	Negative	Total
Predicted Classes	Positive	25	6	31
	Negative	2	21	23
	Total	27	27	54

Table 2: Confusion matrix on test data. Accuracy is 85%.

4.2 Analysis of Line Graphs for Data Extraction

A line graph is a 2D plot with two axes and single or multiple curves in the plotting region. The generic approach for data extraction from such graphs consists of following steps: 1. Identification of axes; 2. Extraction of axes values and 3. Extraction of legend text and associating the curves with the legends. There has been work on text extraction[8] and curve reconstruction[9] indicating that the sub-problems have been explored before, still, no complete algorithm exists for reliable and scalable data extraction.

We have analyzed more than 300 line graphs sampled from a collection of 10,000 computer science papers published in top fifty conferences over a span of 2004 to 2014. Our analysis indicates that two main challenges in this problem are: 1. Curve reconstruction i.e. assigning each point in the plot region to one of the curves and 2. Curve-legend association i.e. assigning each curve to a legend text. It is very hard to solve them for monochrome plots where curves are plotted with black pixels and distinguished by markers or other patterns (see figure 4). The problem is naturally easier for color plots where curves can be distinguished by their color.

52% of plots in our dataset are color plots, i.e. curves are not plotted with only black/ gray pixels. Obviously, a color plot doesn't automatically imply that each curve is plotted with a separate color. However, for most of these plots (89%) that is the case. We identified two other challenges: 1. Overlapping curves (see figure 3) and 2. Non ideal plotting region.

When curves overlap, some foreground pixels might have a color which is a combination of colors from multiple curves. Identifying these base colors is the focus of our ongoing work. Another problem rises from the variation in plotting styles. Ideally, a plotting region should contain only two components: 1. Curves and 2. A legend region. However, that is often not the case. Only 58% of the plots in our dataset had an ideal plotting region. We observed that there are four main reasons for plots being "non ideal": 1. The plotting region had a grid structure (as in figure 4): 87%; 2. Legend region was not present (15%) or not in the plotting region (13%); 3. There are text/ graphic elements in the plotting region which are neither legend nor curve (15%) and 4. Plotting region background is non white (10%). Note that these characteristics are not exclusive, i.e. there are non ideal plotting regions which have a grid structure as well as the legend region is not inside the plotting region. In conclusion, if the grid structures can be removed from the plotting region, most non ideal plots would become ideal.

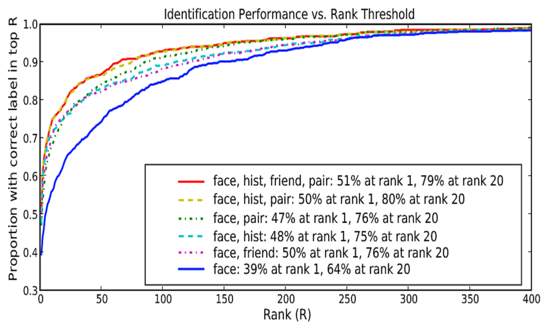


Figure 3: A color plot with overlapping curves.

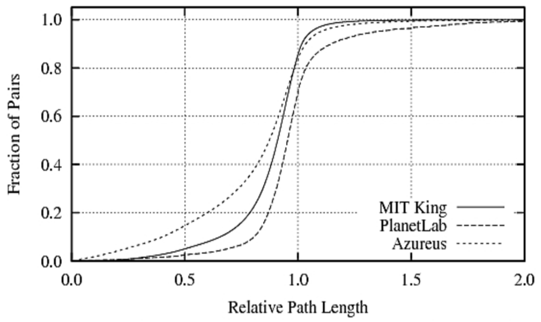


Figure 4: A monochrome plot where the curves can only be separated by their patterns.

To summarize, it is hard to extract data even from color line graphs. But the existing algorithms can be used after following features are incorporated: 1. Removing the "grid" structures from the plotting region and 2. Finding the base colors for pixels where the curves overlap. This analysis is presented to motivate further discussion in this area. Analysis of monochrome line graphs is one of our future works.

5. SEARCH ENGINE WITH SEMI AUTOMATIC DATA EXTRACTION MODULE

Our search engine allows users to search on more than 90,000 figure captions and mentions extracted from chemistry journal articles published by the Royal Society of Chemistry. The search results page provides a list of ranked results (all figures found relevant by our system). Each item in the list corresponds to a link that leads users to a page that shows the figure, caption and mention.

Our system provides two benefits over existing digital libraries, where the users can search on the figure metadata by appending the term "figure" with the search query. Firstly, the user can download the figure for further analysis. Secondly, we have integrated a web based system to facilitate data extraction from the figures. A typical use scenario is the user searches for a query term and follows the link from the SERP to the figure description page, and, if interested, follows a link from there to the data extraction page (see figure 5). The system is a modified version of a publicly available software called WebPlotDigitizer [5]. For 2D graphs, the system takes the following inputs from the user: beginning and ending points of X and Y axes (by recording mouse clicks),

[5]`http://arohatgi.info/WebPlotDigitizer/`

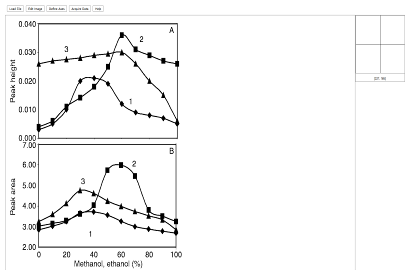


Figure 5: Landing page in the data extraction process.

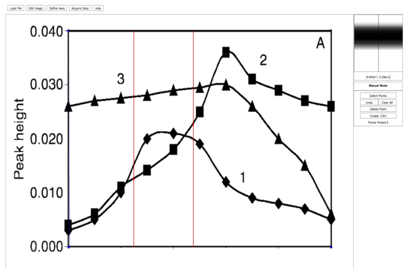


Figure 6: Guiding users in data extraction process. The red line helps to remember the x value for which the user want to extract the y values for different curves.

axis scales (linear/logarithmic) and limiting data values in X and Y axis. If the curves in the image are plotted using separate colors, it is easy to identify them. Binary/grayscale images pose greater challenges. For each point clicked on a curve, we return the actual data value for that point (calculated from the axes locations and their data range). We assume that for line graphs, the user is interested in knowing different Y-values that correspond to different curves for a particular X-value. It might be difficult for the user to click on different (x,y) points on the curves where the x value is same. The system guides the user in that process (see figure 6).

6. CONCLUSIONS AND FUTURE WORK

To the best of our knowledge, we are the first to propose a complete architecture for analysis of figures in scholarly documents. Our architecture contains multiple modules for different tasks. In this paper, we report our work on three modules of our architecture. We report a novel method for extraction of graphics from PDF documents, a simple but effective classifier for extracted figures, and a search engine built using our extractor. Apart from improving our current work on figure extraction, our work in the immediate future involves developing metadata formats for different types of figures, and developing algorithms for fully automatic data extraction from different types of figures. In the future, we plan to use the extracted data and metadata to create a natural language summary of the figures.

7. ACKNOWLEDGEMENTS

We gratefully acknowledge partial support from the National Science Foundation and NPRP grant # 4-029-1-007 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

8. REFERENCES

[1] W. Browuer, S. Kataria, S. Das, P. Mitra, and C. L. Giles. Segregating and extracting overlapping data points in two-dimensional plots. In *Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries*, pages 276–279. ACM, 2008.

[2] S. S. Bukhari, F. Shafait, and T. M. Breuel. Improved document image segmentation algorithm using multiresolution morphology. In *IS&T/SPIE Electronic Imaging*, pages 78740D–78740D. International Society for Optics and Photonics, 2011.

[3] S. Carberry, S. Elzer, and S. Demir. Information graphics: an untapped resource for digital libraries. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 581–588. ACM, 2006.

[4] S. Carberry, S. Elzer, N. Green, K. McCoy, and D. Chester. Understanding information graphics: a discourse-level problem. In *Proceedings of the Fourth SIGDial Workshop on Discourse and Dialogue*, pages 1–12, 2003.

[5] H. Chao and J. Fan. Layout and content extraction for pdf documents. In *Document Analysis Systems VI*, pages 213–224. Springer, 2004.

[6] S. R. Choudhury, P. Mitra, A. Kirk, S. Szep, D. Pellegrino, S. Jones, and C. L. Giles. Figure metadata extraction from digital documents. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 135–139. IEEE, 2013.

[7] A. Coates, A. Y. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, pages 215–223, 2011.

[8] S. Kataria, W. Browuer, P. Mitra, and C. Giles. Automatic extraction of data points and text blocks from 2-dimensional plots in digital documents. In *Proceedings of the 23rd national conference on Artificial intelligence*, volume 2, pages 1169–1174, 2008.

[9] X. Lu, S. Kataria, W. J. Brouwer, J. Z. Wang, P. Mitra, and C. L. Giles. Automated analysis of images in documents for intelligent document search. *IJDAR*, 12(2):65–81, 2009.

[10] G. Nagy and S. Seth. Hierarchical representation of optically scanned documents. In *Proceedings of International Conference on Pattern Recognition*, volume 1, pages 347–349, 1984.

[11] V. Prasad, B. Siddiquie, J. Golbeck, and L. Davis. Classifying computer generated charts. In *Content-Based Multimedia Indexing, 2007. CBMI'07. International Workshop on*, pages 85–92. IEEE, 2007.

[12] M. Savva, N. Kong, A. Chhajta, L. Fei-Fei, M. Agrawala, and J. Heer. Revision: Automated classification, analysis and redesign of chart images. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 393–402. ACM, 2011.