Automated Data Extraction from Scholarly Line Graphs

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

Shuting Wang
Computer Science and Engineering
Pennsylvania State University
sxw327@psu.edu

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

Abstract—Line graphs are ubiquitous in scholarly papers. They are usually generated from a data table and often used to compare performances of various methods. The data in these figures can not be accessed. Manual extraction of this data is hard and not scalable. On the other hand, automated systems for such data extraction task is not yet available. We report an analysis of line graphs to explain the challenges of building a fully automated data extraction system. Next, we describe a system for automated data extraction from color line graphs. Our system has multiple components: image classification for identifying line graphs; text extraction from the figures and curve extraction. For the classification, we show that unsupervised feature learning outperforms traditional low-level image descriptors by 10%. For the text extraction, our heuristics outperforms the accuracy of the previous method by 29%. We also propose a novel curve extraction method that has an average accuracy of 82%. A large partially annotated dataset for future research is described.

I. Introduction

Scholarly papers usually contain many figures and tables. For example, among 10,000 randomly selected articles published in top 50 computer science conferences[1] between 2004 and 2014, more than 70% contained at least one figure, more than 43% contained at least one table, and more than 36% contained at least one figure and one table. While there have been many works about extraction and understanding of tables, figures have received less attention [2].

Previously, Ray Chaudhury et al. [3] explored figure metadata (caption/mention) extraction. A recent paper by Clark et al. [4] has proposed methods for automated extraction of figures from scholarly PDF documents. Using their method we extracted more than 40,000 figures from scholarly papers. This paper focuses on line graphs that are a subset of these figures. Preliminary analysis of the dataset is presented in section IV.

Line graphs are plots with single or multiple curves in the plotting region. These figures are used to compare multiple methods and are generated from data tables. For example, see figure 1, a line graph that was generated from a data table that can not be accessed from the paper. It is extremely beneficial to regenerate that table, but manual methods[2] are tedious and not scalable. A fully automated system doesn't exist till date, and this work aims to bridge that gap.

We analyzed more than hundred randomly selected line graphs from our dataset to understand the specific challenges of building a fully automated data extraction system. From our analysis (section III), we were able to identify easy and hard cases for data extraction. Our current system focuses on easy cases: line graphs where the curves or data points are drawn with separate colors.

The first module in our system is a classifier that classifies an input image as a positive (color line graph) or negative (bar graphs, pie chart, photograph) instance. While multiple features for this problem has been proposed, we are the first to show that unsupervised feature learning outperforms traditional feature descriptors such as SIFT or HoG.

For color line graphs, the generic algorithm for data extraction is:

1) Extract text "words" from the figure.
2) Classify the words as X-axis/Y-axis value/labels or legends. For the data extraction, every point in the plot region needs to be mapped into a (X-value,Y-value) pair. Assuming the plot scales are linear, two pairs of X-axis values and Y-axis values are necessary and sufficient.
3) Extract the curves and associate them with the legends.

Kataria et al. [9] proposed heuristics for text extraction from plot images, but their method was not evaluated using the metrics standardized by document analysis community [8] in recent times. Our system improves on their method.

Next, extracted text is classified in seven classes such as axes value/labels (see section VI-C). It is easy to see that once two X-axis values and two Y-axis values are identified properly, every pixel in the plot region can be mapped to a "data point". The next step is the curve segmentation i.e. assigning each non-text pixel in the plot region to one of the curves. As we consider color plots, it might appear that the process is trivial, and a color based segmentation would suffice. While a color plot usually contains less than ten "visually distinguishable" colors, it might have more than one thousand distinct RGB color values due to anti-aliasing. Our algorithm solves that problem.

II. RelatedWork

Classification of computer generated charts is a well-studied problem, and multiple features have been proposed. Shao et al. [15] and Huang et al. [7] used low-level graphemes extracted from vector graphics. Prasad et al. [12] used a set of very complicated features, extracted through various transformations on the image. Most recent work by Savva et al.

[1] http://academic.research.microsoft.com/
[2] http://arohatgi.info/WebPlotDigitizer/app/

client number	times	
	global locking	object range locking
4	0.005	5
8	0.01	9
12	0.12	11

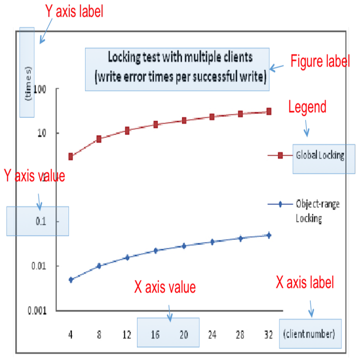

Fig. 1. An example 2D line graph and the data table from which it was generated. It also shows the seven classes of text inside the figure. The mapping between these classes and the data table is apparent.

[13] showed unsupervised feature learning outperforms feature engineering. Our work reinforces that finding (section V).

Early papers by Futrelle [5] introduced the concept of “Generalized Equivalence Relations” (GER) between graphical elements of a figure (symbols, lines, curves) and proposed a pyramidal spatial index for efficient computations of GERs. Subsequent work proposed a more complete grammar for parsing the graphemes [6]. While these works have greatly advanced the understanding of scholarly figures, they did not focus much on the automated data extraction problem itself. For example, they don’t clarify much how these graphemes were extracted from the images, which would be the most important step for the data extraction. In more recent works by Shao et al. [14], [15] these graphemes were extracted from vector graphics embedded in PDFs and not raster graphics. Therefore, it would be hard to generalize their approaches to all figures. Kataria et al. [9] proposed an architecture for automated data extraction from 2D plots and their work is most relevant to ours. A more comprehensive version of their architecture [11] reported curve extraction techniques for line graphs, but only for continuous curves. We explore another aspect of the problem.

III. Analysis of Line Graphs for Data Extraction

A line graph has curves and text words. A word can be classified as one of the following: 1. X-axis value, 2. Y-axis value 3. X-axis label, 4. Y-axis label, 5. Figure label, 6. Legend and 7. Other text. The metadata structure for a line graph is a table as shown in figure 1. Axes values are used to generate the “data value” for the curve pixels. The axes labels are the headers for the columns in the table. For one X-value, there are multiple Y-values, corresponding to multiple curves. Legend texts are the column headers for these curves. The figure label is also a metadata for the graph, along with captions and mentions. An automated system for data extraction would need to extract all these information from the figure. While previous work has explored parts of the problem, they are hard to integrate into an architecture due to the variability of the data.

Our research questions are: 1. What are the variations in the plotting styles that make the text/graphics extraction difficult? And 2. Can we identify easy and hard instances of the problem? To answer them, we analyzed more than hundred line graphs sampled from a large collection (section IV).

Many problems arise from the variations in the plotting styles. Previous work [11] assumes that a line graph can be segmented into X-axis, Y-axis and plotting region. Another assumption is that the plotting region is always “ideal”, i.e., contains only two components: 1. Curves and 2. A legend region. We find that is often not the case. Only 58% of the plots in our dataset had such an ideal plotting region. We observed that there were four main reasons for plots being “nonideal”: 1. The plotting region had a grid structure (as in figure 2): 87%; 2. Legend region was not present (15%) or not in the plotting region (13%); 3. There were text/ graphic elements in the plotting region which were neither legend nor curve (15%) and 4. Plotting region background was nonwhite (10%). Note that these characteristics were not exclusive, i.e. there were nonideal plotting regions that had grid structures, as well as legend regions were not inside the plotting region.

These statistics indicate two problems with existing methods: 1. The “grid” structure and non-white background needs to be removed from the plotting region before applying any algorithm, and 2. The assumption of legend regions being inside the plotting region is not valid.

For curve extraction, line graphs can broadly be classified in two classes: 1. Binary/ grayscale plots where curves are plotted with black/gray pixels and are distinguished by markers or other patterns. Liu et al. [11] proposed methods for this problem, but they assumed that the curves were always connected. That assumption is not valid in most real graphs (see figure 2). 2.Color plotswhere curves can be distinguished by their color. 42% of plots in our dataset are color plots. 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. Naturally, curve extraction is easier for these plots.

To summarize:

1) The hardness of the problem lies in the variation of the plotting styles, and that aspect has gone largely unnoticed in previous works.
2) There are quite a few “easy” instances of the problem that have not been explored properly. These plots are the focus of this paper.

There are three challenges in curve extraction from color plots: 1. Removal of the grid structure and non-white background, 2. Identifying “visually distinguishable colors” and 3. Overlapping curves. Section VI-C reports our methods for these problems. Also, for the text extraction and classification, we do not make any assumption about the legend region being inside the plot region as in the previous work.

IV. Dataset

Our entire dataset consists of 40,000 figures extracted from 10,000 articles published in top fifty computer science conferences between 2004 and 2014. The figures were extracted using a recently released system by Clark et al. known as “pdffigures” [4]. Their system produces a grayscale image file and a JSON metadata file for each figure in the document. The metadata contains following information:

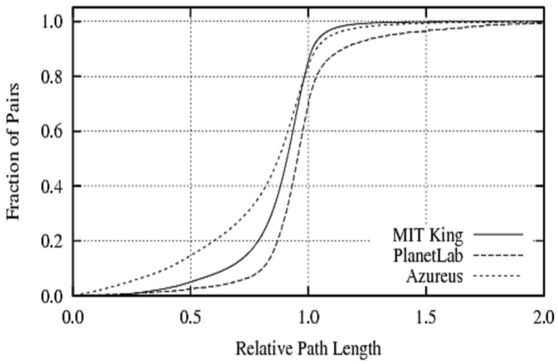

Fig. 2. A monochrome plot (extracted from [10]) where the curves can only be separated by their patterns.

- Page number for the page where the figure appears.
- Location of the figure on the page (bounding box).
- Caption of the figure.
- If the text inside the figure can be extracted from the PDF itself, then for each text word in the figure following metadata is available: 1. The text of the word, 2. The bounding box of the word, and 3. The orientation (horizontal/vertical). Typically, this happens when figures are embedded as vector graphics (eps/ps/PDF).

We made two modifications to the existing system: 1. From the metadata, we re-extracted the figure as a color image. However, around 60% of these images were originally embedded in the PDF as a grayscale image, therefore, were extracted as such. 2. We modified the metadata to include the paragraphs in the document where the figure was mentioned [3].

We manually examined the images and found around 50% of these figures contained sub-figures: often a combination of line graphs, bar charts, and pie charts. From a random selection of 10,000 figures, we manually selected 2250 plots that were either a line graph or contained at least one line graph as a sub-figure. Among them, 882 were color plots. These 882 images were subsequently sampled for various experiments in this paper except for the figure classification experiment (to maintain consistency with previous work). A hundred of these figures were manually tagged with word class labels as discussed in section VI-B). To avoid bias in the algorithm design, a completely separate sample of 120 figures were used for the problem analysis (section III). This dataset will be made publicly available.

V. Classification of Figures

The first step in our architecture is a classification problem: an input image is classified as a line graph or not. Previous works have explored a similar multiclass classification problem, where images are classified as a line graph, a bar graph or a pie chart, etc. Surprisingly, none of the previous methods has used common low-level image descriptors such as SIFT and HoG. We used these descriptors in a bag of words (BoG) model.

The SIFT feature descriptor tries to find key points in an image using scale-space extrema in a difference-of-Gaussians (DoG) pyramid. A Gaussian pyramid is created by repeated smoothing and subsampling of an image by a Gaussian kernel, and the DoG pyramid is created by computing the difference between the adjacent levels in the Gaussian pyramid. Histogram of Oriented Gradients (HOG) algorithm counts occurrences of gradient orientation in localized portions of an image and combines them to produce a final feature descriptor. Both HoG and SIFT have been extensively used in object detection and image classification. Our motivation for using them was to represent specific structures such as bars in the bar graphs and curves in the line graphs in the feature space.

The BoG method in our case works in two steps. First, interest points and descriptors for these points are extracted from the image and then clustered to create a “visual words” dictionary or codebook. Then, each image is represented as a frequency distribution over these visual words. This distribution is used as a feature vector.

For the same classification problem, Savva et al. [13] used random patches of pixels in a BoG model instead of image descriptors. Their method is motivated by recent advances in unsupervised feature learning that has shown excellent promise in natural image classification. We used their dataset for the sake of comparison. In summary, we had 1130 images divided in eight categories: bar charts (215, 19%), line graphs (147 13%), maps (200 17.7%), Pareto charts (117, 10.3%), pie charts (118 10.4%), radar plots (86 7.6%), scatter plots (159 14.2%) and Venn diagrams (88 7.8%). We used stratified k-fold cross validation and compared the accuracies on the test data. The results are shown in table I. In terms of average accuracy (mean of class specific accuracies), HoG descriptor (68%) performed better than the SIFT descriptor (40%) but neither of them outperformed the random patches method (80%). This indicates that the scholarly charts usually do not contain complex structures and randomly selected patches are sufficiently good representations.

	Random (Savva et al. [13])	HoG	SIFT
Curve Plot	73%	64%	48%
Maps	84%	68%	52%
Pareto Plot	85%	67%	51%
Pie Plot	79%	71%	42%
Radar Plot	88%	70%	44%
Scatter Plot	79%	63%	40%
Bar Plot	78%	66%	39%
Table	86%	72%	46%
Venn Plot	75%	68%	42%
Average	80%	68%	40%

TABLE I. Accuracy results for computer generated charts classification: random patch based unsupervised feature learning outperforms low-level image descriptors such as HoG and SIFT.

VI. Word Extraction and Classification

The goal of this module is to extract the text words from the image and classify them to aid the data extraction process.

A. Word Extraction

Text extraction in our case is easier than natural scene images. One choice is to use off-the-shelf OCR systems such as Tesseract, but they have less accuracy on these images because of the text sparsity [9], [17]. Zhu et al. [17] used a convolutional K-means approach to extract text regions from patent images. While their method achieved good accuracy (71%), it