Scalable Algorithms for Scholarly Figure Mining and Semantics

Sagnik Ray Choudhury (sagnik@psu.edu)
Shuting Wang (sxw327@psu.edu)
C. Lee. Giles (giles@ist.psu.edu)
Pennsylvania State University
CiteSeerX and the Scholarly Semantic Web

- CiteSeerX (http://citeseerx.ist.psu.edu)
  - Largest collection of full text scholarly papers freely available on the Web (7M and growing)
  - Provides full text and citations search (upcoming: table and figure search)

- Semantics in CiteSeerX (more on this in the next talk):
  - Understanding document type (paper/resume)
  - Extraction and disambiguation of scholarly metadata (title, author, affiliation)
  - Information extraction from tables and figures in scholarly PDFs.

- This presentation:
  - A modular architecture for analysis of scholarly figures.
  - Each module generates a “searchable metadata” for a figure.
  - New algorithms, scalability improvement over existing ones.
Motivation

• Most scholarly documents contain at least one figure – many millions of figures.
• Figures are used to for many purposes. Data in such figures is invaluable for much research
• Experimental figures contain data that is NOT available in the document and sometimes nowhere else.
• We can automatically
  • Find and extract figures
  • Extract data from some figures
• With that data, experimental figures (and tables) can be reduced to facts–> <problem (key phrase extraction), experimental method (TextRank), evaluation metric (precision, recall), dataset (InSpec), result(32%)>

Precision-recall curves for unsupervised methods in key phrase extraction

There are five precision recall curves (singlerank ..) in this figure.

precision reduces as recall increases. <singlerank>
precision increases as recall increases. </singlerank>

precision increases as recall increases.</textrank>

singlerank, singlerank+ws=2, singleank+unweighted curves are similar and higher than the last two.
</overalltrend>
</description>
• On a sample of 10,000 CS articles, 69.85% contains figures, 43.03% contains tables and 35.90% contains both figure and tables.
• Figures are embedded in PDF in raster graphics format (JPEG/ PNG) or vector graphics format (PS/EPS/SVG). 70% of all 40,000 figures in our dataset were embedded as vector graphics. They should be extracted and processed as such.
Related Work

• Scholarly figures have received less attention than scholarly tables [10].
• Two directions of information graphics research:
  • NLP: Understanding the intended message of the figures (line graphs [9], bar charts [11].)
    • Not much discussion on the extraction of data from figures.
    • Dataset is not scholarly figures but images from the Web. Easier to understand.
  • Vision: Data extraction from 2D plots [7,8].
    • Extracted and analyzed raster graphics, whereas in many domains including computer science, most figures are embedded as vector graphics.
    • Results were reported on synthetic data.
• Closest to our work is DiagramFlyer in University of Michigan[12]
  • Doesn’t distinguish between compound and non compound figures.
  • Doesn’t understand the type of the figure (line graph/ bar graph/ pie chart)
  • Doesn’t extract data from figures.
Figure and Table Extraction

• Previous work: machine learning based figure and metadata extraction[1,2]

• *Pdffigures* figure extraction tool by Clark et al.[3]
  • Fast (processed 6.7 Million papers in around 14 days parallelized on a 8 core machine.) and *mostly* accurate, in C++. Available at https://github.com/allenai/pdffigures
  • A newer version reported recently at JCDL 16.

• Produces a low resolution BW raster image for the figure and a JSON file with caption, and the text inside the figure (if the figure was embedded in a vector graphics format)

• We rewrote it in Scala to integrate with the JVM based extraction architecture of CiteSeerX (https://github.com/sagnik/pdffigures-scala)
Compound Figure Detection

• Binary classification: a figure is compound (contains sub figures) or not (around 50%).
• Motivation: Compound figures need to be segmented before processing.
• Detection is relatively easy, segmentation is hard[4]
• 300 SIFT features and presence of a white line spanning the image.
• Textual features: BoW from captions + delimiters (‘(a)’, ‘i.’)
• Linear kernel SVM -> 85% accuracy with Less than 1 second per image.
  • https://github.com/sagnik/compoundfiguredetection
• If compound figure, produce metadata 2: (caption, mention, words)
• If non compound-> classify as line graph, bar graph or others. If others, produce metadata 2.
Figure Classification

• SIFT features are bad for this task, random patches are better\cite{5}.
  • Offline step: Create a dictionary of 200 words by taking random patches from a separate subset of training data.
  • For each pixel in a image (training+test) extract a patch and produce a 200 bit vector, all zeros except one, the index of the closest word ($l_2$ distance) in the dictionary.
  • Sum the vectors over quadrants and concatenate: 800 bit vectors.
  • 83% F1-score using linear kernel SVM. But, takes 92 seconds per image due to the dense sampling step.

• Two approaches for scalability improvement:
  • Randomly sample 1000 pixels instead of all pixels. Time improvement: 15 times. F1-score reduces by 6%.
  • Instead of Euclidian distance, use cosine distance after normalizing both the dictionary and the image. Cosine and Euclidian distance are the same for unit vectors.
  • Problem reduces to matrix multiplication + finding out the index of the max value.
  • Time improvement : 15 times, F1-score unchanged.
Figure Text Classification

• With “metadata 3” We want to make SQL like queries (\texttt{x_axis_label}: precision AND \texttt{y_axis_label}: recall AND \texttt{legend}: SVM AND \texttt{caption}: dataset).

• Text from figure is classified in seven classes: axes values and labels, legend, figure label and other text.

• Input features are based on the text of a “word”, location and orientation.

• Distance from boundary, number of words in the vicinity and more.

• 4400 words from 165 images were manually tagged.

• Five fold stratified cross validation: random forest with 100 decision trees has more than 90% accuracy for all classes except one.

• Only text based features: classification takes less than a second per image.
  • \url{https://github.com/sagnik/figure-text-classification}
Final Metadata: Natural Language Summary for a Line Graph

Summary
This plot shows Precision (%) vs Recall (%) curves for following methods: 1. Te SingleRank.

The curve trends are:
- Curve TextRank has increasing trend
- Curve SingleRank+Window size=2 has decreasing trend
- Curve SingleRank+Unweighted has decreasing trend
- Curve SingleRank has decreasing trend

The X axis values are: 20,40,60,80,100
The Y axis values are: 50,40,30,20,10,0,0

- Original figure extracted from Hassan and Ng.[6].
- Precision-Recall curves for different methods in “unsupervised key phrase extraction” on InSpec dataset.
- For more details, see http://personal.psu.edu/szr163/hassan/hassan-Figure-2.html
**Natural Language Summary for a Line Graph**

- **Steps**: curve extraction, curve trend identification and legend curve mapping.

- **Previous work**[7,8,9] in curve extraction from line graphs has always considered raster graphics.
  - Before 2015[2,3], there was not any batch extractor for figures embedded as vector graphics.
  - Both these methods find out the bounding box of a figure, rasterizes the PDF page with a low resolution and crops off the region.

- **Our contribution**: Extract the figures in scalable vector graphics (SVG) format if they were embedded as a vector graphics.

- **Curve extraction** is both accurate and fast for vector graphics.
Extracting Figures in SVG Format: Motivations

• Need at least 70 ppi image for image processing based analysis of figures, PDF rasterization takes 50-60 seconds on a desktop.

• For color curves it is relatively easier to separate pixels from a high resolution image. Overlapping curves pose serious problem.

• For black and white curves the problem is naturally harder.

• SVG images have paths (text commands), instead of pixels.

• A “curve” in an SVG image is a collection of paths.

• Each path has a color attribute.

• Paths can be clustered based on their color just using regular expressions. Each such cluster is a curve.

• These SVG images can be produced in 4-5 seconds.
SVG Figure Extraction

• Convert the PDF page in SVG using off the shelf tools: InkScape.
  • [http://personal.psu.edu/szr163/svgconversionresults/converted.html](http://personal.psu.edu/szr163/svgconversionresults/converted.html)

• Find bounding box of each path and character; output the ones within the bounding box of a figure.

• Problems:
  • A path has multiple commands (draw line, Bezier curve), each with a sequence of arguments.
  • `<m 20,30 40,0 0,40 z>` draws a rectangle, but that’s not apparent.
  • Many paths are grouped under a grouping element, groups are grouped further: nested hierarchical structure, same with the text.

• Solution:
  • Developed an SVG parser that reduces any path to an “atomic” representation: has no group, exactly one command with one argument and a bounding box.
  • Available at [https://github.com/sagnik/inkscape-svg-processing](https://github.com/sagnik/inkscape-svg-processing).
Curve Legend Association and Natural Language Summary

• Evaluation is visual: a curve is considered correctly extracted if at least 90% of the curve can be seen and at most 10% of any other curve can be seen.

• Precision and recall for color curves is 90.08% and 88% on 200 plots:
  • Black curves are not extracted.
  • Grid lines drawn in gray are extracted as curve.

• Curve legend association: rasterization, then bipartite matching.
  • Cost function between a curve C and a legend L as the horizontal distance between L and the pixel from C closest to L.
  • If no pixel from the curve exists within a rectangle of width 20 to the left or right of the legend, the cost is infinity.
  • Minimize total cost of assignment.
  • Precision is 81%, error is due to “wrongly” extracted curves.

• Natural language summary is generated using the change in gradient of the curves.
Summary and Future Work

• A modular architecture for understanding the semantics of scholarly figures.
  • Generate searchable metadata in increasing order of information richness.
• Algorithms are improved for scalability and accuracy.

• Extended work: extract BW curves (https://github.com/sagnik/linegraph-curve-separation)
• Improve the scalability of SVG extraction: Ongoing work, initial results: < 1s.
• Generate a publicly available data set of several million figures.
Acknowledgements

• Anonymous reviewers for the suggestions.
• Dr. Sven Groppe for preparing the camera ready version.
• National Science Foundation and Qatar Foundation for support.
• Jian, Kyle and Rabah for helpful discussions.
References


References


