PDFFigures 2.0: Mining Figures from Research Papers

Christopher Clark Santosh Divvala

Allen Institute for Artificial Intelligence
University of Washington
{chrisc, santoshd}@allenai.org

http://pdffigures2.allenai.org

ABSTRACT

Figures and tables are key sources of information in many scholarly documents. However, current academic search engines do not make use of figures and tables when semantically parsing documents or presenting document summaries to users. To facilitate these applications we develop an algorithm that extracts figures, tables, and captions from documents called "PDFFigures 2.0." Our proposed approach analyzes the structure of individual pages by detecting captions, graphical elements, and chunks of body text, and then locates figures and tables by reasoning about the empty regions within that text. To evaluate our work, we introduce a new dataset of computer science papers, along with ground truth labels for the locations of the figures, tables, and captions within them. Our algorithm achieves impressive results (94% precision at 90% recall) on this dataset surpassing previous state of the art. Further, we show how our framework was used to extract figures from a corpus of over one million papers, and how the resulting extractions were integrated into the user interface of a smart academic search engine, Semantic Scholar (www.semanticscholar.org). Finally, we present results of exploratory data analysis completed on the extracted figures as well as an extension of our method for the task of section title extraction. We release our dataset and code on our project webpage for enabling future research (http://pdffigures2.allenai.org).

Keywords

Scalable figure extraction; academic search engine; section title extraction; figure usage analysis

1. INTRODUCTION

Traditional tools for organizing and presenting digital libraries only make use of the text of the documents they index. Focusing exclusively on text, however, comes at a price because in many domains much of the important content is contained within figures and tables. Especially in scholarly domains, authors frequently use figures and tables to compare their work to previous work, to convey the quantitative results of their experiments, or to provide graphics that help readers understand their methods. Therefore parsing figures and tables is a necessary component of any system that seeks to gain a semantic understanding of such documents.

Tables and figures also have the potential to be used as powerful document summarization tools. It is common to get the gist of a paper by glancing through the figures[1], which often contain both the main results as well as visual aids that outline the work being discussed. Being able to extract these figures and present them to a user would be an effective way to let users quickly get an overview of the paper's content. To this end, we introduce PDFFigures 2.0. PDFFigures 2.0 takes as input computer science papers in PDF format and outputs the figures, tables, and captions contained within them.

Our work builds upon the PDFFigures algorithm [5]. The approach used by [5] has high accuracy but was only tested on papers from a narrow range of sources. In this work, we improve upon that method to build a figure extractor that is suitable for use as part of academic search engines for computer science papers. To meet this goal we improve upon the accuracy of PDFFigures [5] and, more importantly, build an extractor that is effective across the entire range of content in a digital library. This requires an approach that is robust to the large number of possible formats and styles papers might use. Particular challenges include handling documents with widely differing spacing conventions, avoiding false positives while maintaining the ability to extract a broad range of possible captions, and extracting a highly varied selection of figures and tables.

Our approach follows the same general structure used in [5] (see Section 3) and employs data-driven heuristics that leverage formatting conventions used consistently in the computer science domain. Following a heuristic approach makes our method transparent and easy to modify [13], which we have found to be important for developing an effective solution to this task.

While our focus is on extracting figures, our method also produces a rich decomposition of the document and analysis of the text. In this paper we demonstrate how this analysis can be leveraged for other extraction tasks, such as identifying section titles. Section titles are important because they reveal the overall structure of the document, and can be a crucial feature for upstream components analyzing

[1]Throughout this paper we use the term "figures" to refer to both tables and figures along with their associated captions

JCDL '16, June 19-23, 2016, Newark, NJ, USA

DOI: http://dx.doi.org/10.1145/2910896.2910904

body text. Section titles can also be used to identify which section figures were introduced in, thereby providing some additional context for interpreting extracted figures. We evaluate our section title extraction method on a dataset of over 50 papers and compare our results against prior work.

In order to evaluate PDFFigures 2.0 against a diverse set of documents, we introduce a new dataset of over 325 computer science papers along with ground truth labels for the locations of figures, tables, and captions within them. We also show how our method was used to extract figures from over one million documents and integrated into the user interface for Semantic Scholar [3], a smart academic search engine for computer science papers. We conclude by using this dataset to study how figure usage has evolved over time, how figure usage relates to future citations, and how figure usage differs between conference venues.

2. RELATED WORK

For a comprehensive survey of previous work in figure extraction as well as relevant open source tools, please see [5]. In this section we review some recent developments in the field as well as exciting applications of figure extraction.

A machine learning based approach to figure extraction was recently proposed in [11]. Their method classifies the graphical elements in a PDF as being part of a figure or not. Elements that were classified as being part of a figure were clustered to locate individual figures in the document. Rather then working primarily with the graphical elements in a document, our approach focuses on identifying body text and then using layout analysis to locate the figures, which allows our approach to not only extract a wide variety of figures but also generalize to extracting tables.

The possibility of being able to semantically parse figures is an exciting area of research, and the figure extraction method of [5] has already demonstrated its ability to facilitate pioneering work in this area. In [12], researchers experimented with an approach to extracting data from line plots. Given a figure, their system uses a classifier to determine whether the figure is a line plot. If the figure is determined to be a line plot, a word recognition system is then used to locate text in the plot and classify that text as being part of an axis, a title, or a legend. Finally, heuristics based on color were used to identify curves in the plot and match them against the plot's legend. Their work used PDFFigures [5] to extract a large corpus of figures from papers published in top computer science conferences. The figures were mined to collect real world examples of line plots. Since PDFFigures can additionally extract the text contained in vector graphic based figures, these figures were also used to provide ground truth labels for the word detection system. Another recent project has similarly found that figures extracted by PDFFigures can be used to generate large amounts of text detection training data for a neural network [4].

Researchers in [14] introduced a novel framework for parsing result figures in research papers. They used PDFFigures [5] to extract figures from computer science papers and subsequently used a classifier to determine the figure type. For line plots composed of vector graphics, heuristics were used to locate key elements of the charts, including the axis, axis labels, numeric scales, and legend. Apprenticeship learning was then used to train a model to identify the lines, and thus the data, contained within the plot. In all these cases PDFFigures provided the critical building block needed for building tools that are effective on real world figures and papers.

PDFFigures [5] has also been used as a component of PDFMEF, a knowledge extraction framework that extracts a wide variety of entities and semantic information from scholarly documents [15]. In PDFMEF, PDFFigures was used to add figures and tables to the elements PDFMEF is capable of extracting. The authors remarked that PDFFigures is notable for its accuracy and its ability to extract both figures and tables, and concluded by stating "...it[PDFFigures] is arguably one of the best open source figure/table extraction tools."

These projects suggest that the ability to extract figures from arbitrary documents is extremely valuable. With PDFFigures 2.0, we hope to provide a higher quality, more robust tool for researchers wishing to use figures in their work.

The problem of locating section titles within documents has also received attention from researchers, and is addressed in systems such as ParsCit [6], Grobid [8] and SectLabel [9]. All these approaches use machine learning to classify lines of text as being a section title or not. However, we have found that exploiting some natural properties of section titles, such as their use of salient fonts and their location relative to the rest of the document's text, makes heuristic approaches very effective for this task.

3. APPROACH OF PDFFIGURES [5]

Since our work builds upon PDFFigures [5], we review the general strategy employed by [5] in this section. The approach is to focus primarily on identifying the captions and the body text of a document, since these elements are often the easiest to detect in scholarly articles. Once the captions and body text have been identified areas containing figures can be found by locating rectangular regions of the document that are adjacent to captions and do not contain body text. PDFFigures has three phases: Caption Detection, Region Identification, and Figure Assignment.

Caption Detection.

This phase of the algorithm identifies words that mark the beginning of captions within the document. Text is extracted from the document using Poppler [2], and a keyword search is used to identify phrases that are likely to start a caption. False positives are then removed using a consistency assumption: that authors have labelled their figures in a consistent manner as is required by most academic venues.

If the first pass yields multiple phrases referring to the same figure, for example, two phrases of the form "Figure 1", it is assumed that all but one of those phrases is a false positive. If such false positives are detected, an attempt is made to remove them by applying a "filter" that removes all phrases that do not follow a particular formatting convention. Filters are only applied if they do not remove all phrases referring to a particular figure. Filters include (I): Select only phrases that end with a period. (II): Select only phrases that end with a semicolon. (III): Select only phrases that have bold font. (IV): Select only phrases that have italic font. (V): Select only phrases that have a different font size than the words that follow them. Filters are iteratively applied until no false positives are left. If false positives remain but no filter can be applied they fall back on selecting only phrases that start paragraphs, as judged by Poppler's text extraction system.

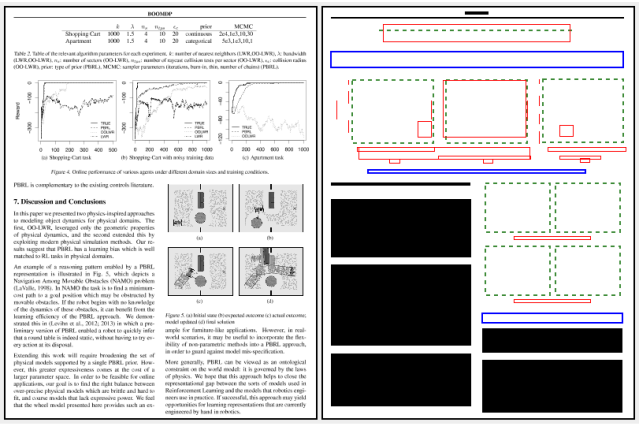


Figure 1: A document page (left panel, from [Scholz et al., ICML 2014]) decomposed into a set of classified regions (right panel). Body text regions are shown as filled boxes, captions and figure text regions as box outlines, and graphical element regions as dashed box outlines.

Region Identification.

Region identification decomposes document pages into regions, each one labelled as either caption, graphical element, body text, or figure text. Caption regions are built by starting from the caption phrases found in the prior step, and combining them with subsequent lines of text. The rest of the text in the document is grouped into paragraphs using Poppler's paragraph grouping mechanism. Paragraphs that are either too large or aligned to the left margin of a column are classified as body text, otherwise they are classified as figure text.

Page headers and page numbers are handled as special cases. PDFFigures checks if pages in the document are consistently headed by the same phrase, and if so marks those phrases as body text. Likewise page numbers are detected by checking if all pages end with a number, and if so marking those numbers as body text.

Finally, the graphical elements of the document are located. To do this each page is rendered as a 2D image using a customized PDF renderer that ignores text. The bounding boxes of the connected components in the resulting image are then used as graphical regions of the document. An example of such a decomposition is shown in Figure 1.

Figure Assignment.

The last step is to assign each caption a region of the document containing the figure it refers to. First, up to four "proposal" regions are generated for each caption. Proposal regions are built by generating a rectangular region adjacent to each side of the caption, and then maximally expanding those regions as long as they do not overflow the page margin, overlap with body text, or overlap a caption. This is shown in Figure 2. For two-column papers regions are constrained to not cross the center of the page unless the caption itself spans both columns.

Next, a single proposed figure region is selected for each caption. To do this a scoring function is used to rate each proposed region based on how likely it is to contain a figure.

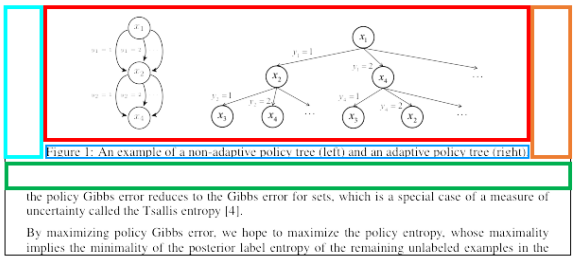


Figure 2: Generating possible figure regions for a caption. For each caption (center blue box), up to four regions are generated as possible figures by maximally expanding boxes that are adjacent to one side of the caption. Page from [Cuong et al., NIPS 2013]

The scoring function gives higher scores to regions that are large and contain graphical elements. To ensure captions are not assigned regions that overlap, they iterate through every possible permutation of how figure regions could be matched to captions. Each permutation is scored based on the sum of the scores of the proposed regions it includes, with regions that overlap given a score of 0. The highest scoring permutation is then selected as the final set of figure regions to return.

An additional complication comes from cases where figures are immediately adjacent, so that they are not separated by any intervening body text or captions. In these cases, proposal regions might get overly expanded and therefore contain multiple figures. To handle these cases, when iterating through permutations, if two proposal regions overlap an attempt is made to split them by detecting a central band of whitespace that separates them. An example of such a figure can be found in Figure 4, second row right column.

4. PROPOSED APPROACH

Our approach builds upon [5] by making crucial updates to its important components. Most of the updates are designed to allow PDFFigures 2.0 to generalize across a wider variety of paper formats. The PDFBox [1] library is used for parsing PDFs.

4.1 Caption Detection

We improve the keyword search of [5] that identifies phrases that might start captions to be effective against more kinds of papers by using a considerably expanded set of keywords. However, naively increasing the number of keywords also increases the number of false positives for each paper. We resolve this problem by adding a number of additional filters to the ones used in [5], such as (I): Select phrases that are all caps. (II): Select phrases that are abbreviated. (III): Select phrases that occupy a single line. (IV): Select phrases that do not use the most commonly used font in the document. (V): Select phrases that are left aligned to the text beneath them. The last filter serves as a general purpose filter for detecting indented paragraphs or bullet points that start by mentioning a figure.

4.2 Region Identification

Our region identification method decomposes each page into caption, body text, figure text and graphical regions as

done in [5].

4.2.1 Caption Region Identification

We use a specialized procedure to identify complete captions once the starting line of that caption has been identified. To make our approach robust to the document's choice of line spacing, we compute the median space between lines in the document. Then, for each mention, we construct the caption by adding lines following the mention that are less than the document's median line space away from each other. This works well in many cases, but can fail on documents where captions have been tightly packed against the following text. To add robustness to this problem, we additionally check to make sure new lines have a similar justification to the lines accumulated so far. We also avoid adding lines of text that overlap a graphic region, or lines of text that are of a different font than the previous lines.

4.2.2 Text Classification

Text classification is the process of determining whether blocks of text on each page should be labelled as body text or figure text. Text classification is made difficult by the wide variety of ways text can appear in figures and was a relatively large source of error in PDFFigures [5]. We develop a new set of heuristics to achieve high performance on this task across many kinds of documents. We leverage the insight that the majority of text in a document is body text, and that body text in a document will have a consistent format throughout the document. As a result, text that is formatted in an anomalous way can be assumed to be figure text. We determine the most common font and font size used in the document, the most common line width used, the most common distance between lines and distance between words, and the most common left margins. These statistics are then used in the following heuristics:

1. Graphic Overlap: Text that overlaps a graphic region is classified as figure text.
2. Vertical Text: Text that has a vertical orientation is marked as figure text.
3. Wide-Spaced Text: Text blocks with above median space between its words are marked as figure text. This heuristic is effective for detecting text in tables.
4. Line Width: Text blocks that are several lines long and of the same width as the most commonly used line width in a document are classified as document text.
5. Small Font: Text that is smaller than the most common font size is classified as figure text.
6. Section Titles: Text that is aligned to a margin or centered, starts with a number or is capitalized, and is of a non-standard font or font size, is marked as body text. This heuristic serves to detect section titles, and forms the basis of our section title extraction method (please see Section 7).
7. Margin Alignment: Text that is aligned to a left margin is classified as body text and any remaining text is classified as figure text.

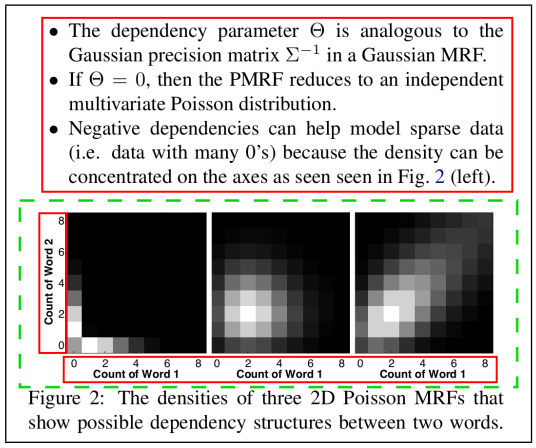


Figure 2: The densities of three 2D Poisson MRFs that show possible dependency structures between two words.

Figure 3: Using clustering to identify figure regions. In this page from [Inouye et al., ICML 2014] the bullet points were mistakenly misclassified as figure text (box outlines). However clustering elements around the Figure's graphics ensures the bullets are not included in the proposed figure region (dashed line).

Generalizing to a wide variety of PDFs also requires a more general page header detection method. We have to handle page headers with inconsistent text, for example page headers that alternate between stating the paper's title and the authors' names, and multiline page headers. PDFFigures 2.0 scans through the first several lines of text on each page. If these lines start above any other text by a sufficiently large margin and appear at the same height on each page they are marked as body text.

4.2.3 Graphical Region Identification

PDFFigures 2.0 locates graphical regions of the document by directly parsing the PDF. Internally PDFs encode graphical elements through the use of various "operators" that draw elements on the page. Operators can be used to draw shapes and curves of different colors or render images embedded in the PDF onto the page. PDFFigures 2.0 scans the PDF and, for each such operator, records the bounding box of the element that operator would draw. To achieve this we make use of functionality introduced in PDFBox 2.0 that provides high level descriptors of the graphical elements being drawn by each operator. The bounding boxes found across the entire page are then clustered by merging nearby bounding boxes, and the resulting merged boxes are used as graphical regions. This approach is much faster than the one used in [5] since it does not require rendering the PDF to a bitmap. This is one of the primary reasons PDFFigures 2.0 is faster than PDFFigures [5] when it comes to locating figures (see Section 6.2).

4.3 Figure Assignment

The final step in our algorithm is to determine the figure regions to return. Possible figure regions are generated for each caption as described in Section 3 (Region Identification). Taking inspiration from [10], where it was found that clustering elements around large graphical regions was an effective way to detect figures, we add a clustering subcomponent to prune the proposed regions. For each proposed

figure region, we check to see if a large graphical region is contained within it. If such a region is found, we cluster elements around that graphical region and then remove elements that were not in the cluster. The motivation for this technique is that if a prominent graphical element, like an embedded image or a chart, is part of a figure it is usually the main focus of that figure. Therefore text that is not near that element is unlikely to be part of that figure. This method helps us to be robust to text that was misclassified, for an example see Figure 3.

We make two additional improvements. First, we found that removing proposed figure regions that partially intersect a word in the document was an effective way to remove many incorrect proposals. Second, the region splitting criteria was adjusted to trigger more frequently and prefer splitting regions across the largest band of whitespace rather than the most central band. This allows PDFFigures 2.0 to extract figures that are adjacent to each other in more cases, but also introduces false positives where figures containing whitespace were incorrectly split between nearby captions. We resolve this issue by adjusting our region scoring function to downweight figure regions that were produced using this new splitting procedure.

5. DATASET

We evaluate our approach on two datasets of computer science papers. The first, which we call the "CS-150" dataset, consists of 150 papers from well-known computer science conferences, introduced by [5]. It is composed of 50 papers from NIPS 2008-2013, 50 from ICML 2009-2014, and 50 from AAAI 2009-2014 with 10 papers selected at random from each conference and year. There are 458 labelled figures and 191 labelled tables. This dataset contains papers that are typical of what researchers read, but because it was only drawn from three conferences it does not capture the full range of styles computer science papers have. In order to test our method on a more diverse set of documents, we gather another dataset by randomly sampling papers used by Semantic Scholar [3] that have at least 9 citations and were published after the year 1999. This dataset will be referred to as the "CS-Large" dataset. We used citation and year restrictions to make this data sample more representative of the kinds of documents researchers encounter in search engines. For these sampled papers, we only annotate (and test on) half the pages selected at random, which allowed us to label twice as many papers. We only label up to 9 pages per paper to ensure longer papers do not contribute an overly large portion of the labelled figures and tables. In total, we annotate 346 papers that originate from over 200 different venues. There are 952 labelled figures and 282 labelled tables. Both datasets were annotated by having annotators mark bounding regions for each caption, figure and table using an image annotation tool. These regions were then cropped to the foreground pixels of the page they were marked on, and the cropped regions were used as ground truth labels for evaluation.

6. EXPERIMENTAL RESULTS

6.1 Figure Extraction

We evaluate PDFFigures 2.0 on both datasets and compare its performance to PDFFigures [5], and a figure extraction method developed for the High Energy Physics domain from [10]. Some of the modifications made to caption detection (see Section 4.1) were ported to PDFFigures, otherwise PDFFigures had a greatly reduced score due to failing to detect certain kinds of captions. Each extractor is expected to return a set of figures for each document. Each returned figure is expected to include a page number, a bounding box for both the figure and its caption, the identifier of that figure (e.g., "Figure 1" or "Table 3"), and optionally the caption text.

Dataset	Extractor	P	R	F1
CS-150	PDFFigures 2.0 (Ours)	0.980	0.961	0.970
	PDFFigures [5]	0.961	0.911	0.935
	PDFPlots [10]	0.624	0.500	0.555
CS-Large	PDFFigures 2.0 (Ours)	0.936	0.897	0.916
	PDFFigures [5]	0.797	0.693	0.741
	PDFPlots [10]	0.678	0.546	0.605

Table 1: Precision (P) and recall (R) on figure extraction.

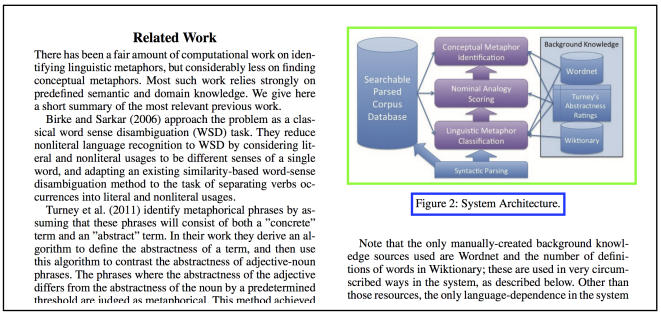
Dataset	Extractor	P	R	F1
CS-150	PDFFigures 2.0 (Ours)	0.979	0.963	0.971
	PDFFigures [5]	0.962	0.921	0.941
	PDFPlots [10]	0.429	0.363	0.393
CS-Large	PDFFigures 2.0 (Ours)	0.932	0.918	0.925
	PDFFigures [5]	0.804	0.563	0.663
	PDFPlots [10]	0.373	0.317	0.343

Table 2: Precision (P) and recall (R) on table extraction.

We follow the evaluation scheme of [5]. Extracted figures with identifiers that did not exist in the hand built labels, or with incorrect page numbers, are considered incorrect. Otherwise a figure is judged by comparing its bounding box against the ground truth using the overlap score criterion from [7]. A bounding box's score is its area of intersection with the ground truth bounding box divided by its area of union with the ground truth bounding box. We consider a bounding box correct if its overlap score exceeds a threshold of 0.80, otherwise it is marked as incorrect.

Captions are evaluated by comparing the returned bounding boxes with the ground truth using the same overlap criterion, although we also consider captions correct if the text extracted from the ground truth region matches the text returned by the extractor. We use this dual criterion for captions because small differences in the bounding boxes of text extracted from the PDF can sometimes cause correct extractions to be judged as errors by the overlap criterion. Figure regions are usually larger and contain graphical elements, and so are not affected by this problem. We consider an extraction to be correct if both the caption and figure region are correct.

Our approach achieves the highest F1 scores on each dataset for both figures and tables. Relative to [5], PDFFigures 2.0 achieves a 3% absolute gain in F1 on the CS-150 dataset. On the more diverse and challenging CS-Large dataset, we see a much larger gain of over 26.2%. The algorithm by [10], although achieving high results in the domain of high energy physics, did not generalize well to the domain of computer science papers.



Related Work

There has been a fair amount of computational work on identifying linguistic metaphors, but considerably less on finding conceptual metaphors. Most such work relies strongly on predefined semantic and domain knowledge. We give here a short summary of the most relevant previous work.

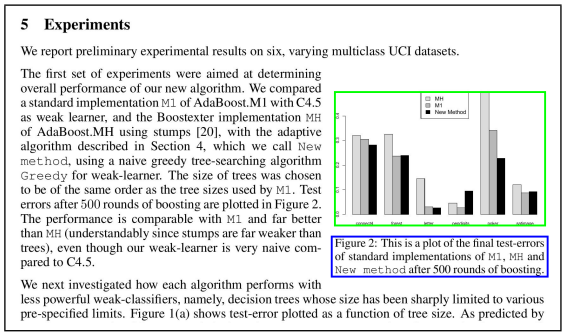
Birke and Sarkar (2006) approach the problem as a classical word sense disambiguation (WSD) task. They reduce nonliteral language recognition to WSD by considering literal and nonliteral usages to be different senses of a single word, and adapting an existing similarity-based word-sense disambiguation method to the task of separating verbs occurrences into literal and nonliteral usages.

Turney et al. (2011) identify metaphorical phrases by assuming that these phrases will consist of both a "concrete" term and an "abstract" term. In their work they derive an algorithm to define the abstractness of a term, and then use this algorithm to contrast the abstractness of adjective-noun phrases. The phrases where the abstractness of the adjective differs from the abstractness of the noun by a predetermined threshold are judged as metaphorical. This method achieved

Figure 2: System Architecture.

Note that the only manually-created background knowledge sources used are Wordnet and the number of definitions of words in Wiktionary; these are used in very circumscribed ways in the system, as described below. Other than those resources, the only language-dependence in the system

Correct extraction of a figure from [Gandy et al., AAAI 2013]



5 Experiments

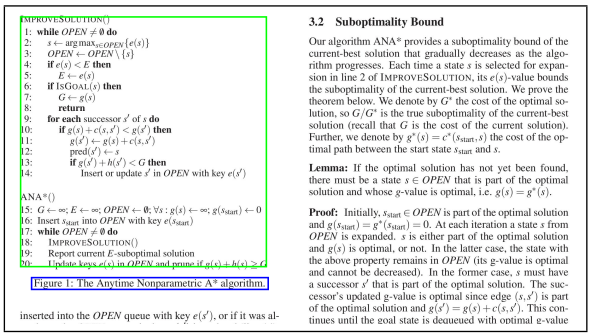
We report preliminary experimental results on six, varying multiclass UCI datasets.

The first set of experiments were aimed at determining overall performance of our new algorithm. We compared a standard implementation `M1` of AdaBoost.M1 with C4.5 as weak learner, and the Boostexter implementation `MH` of AdaBoost.MH using stumps [20], with the adaptive algorithm described in Section 4, which we call `New method`, using a naive greedy tree-searching algorithm `Greedy` for weak-learner. The size of trees was chosen to be of the same order as the tree sizes used by `M1`. Test errors after 500 rounds of boosting are plotted in Figure 2. The performance is comparable with `M1` and far better than `MH` (understandably since stumps are far weaker than trees), even though our weak-learner is very naive compared to C4.5.

Figure 2: This is a plot of the final test-errors of standard implementations of `M1`, `MH` and `New method` after 500 rounds of boosting.

We next investigated how each algorithm performs with less powerful weak-classifiers, namely, decision trees whose size has been sharply limited to various pre-specified limits. Figure 1(a) shows test-error plotted as a function of tree size. As predicted by

Correct extraction of a figure embedded in text from [Mukherjee et al., NIPS 2010]



IMPROVESOLUTION()
1: **while** $OPEN \neq \emptyset$ **do**
2: $s \leftarrow \arg\max_{s \in OPEN}\{e(s)\}$
3: $OPEN \leftarrow OPEN \setminus \{s\}$
4: **if** $e(s) < E$ **then**
5: $E \leftarrow e(s)$
6: **if** ISGOAL(s) **then**
7: $G \leftarrow g(s)$
8: **return**
9: **for each** successor s' of s **do**
10: **if** $g(s) + c(s,s') < g(s')$ **then**
11: $g(s') \leftarrow g(s) + c(s,s')$
12: pred(s') $\leftarrow s$
13: **if** $g(s') + h(s') < G$ **then**
14: Insert or update s' in *OPEN* with key $e(s')$

ANA*()
15: $G \leftarrow \infty$; $E \leftarrow \infty$; $OPEN \leftarrow \emptyset$; $\forall s : g(s) \leftarrow \infty$; $g(s_{\text{start}}) \leftarrow 0$
16: Insert s_{start} into *OPEN* with key $e(s_{\text{start}})$
17: **while** $OPEN \neq \emptyset$ **do**
18: IMPROVESOLUTION()
19: Report current E-suboptimal solution
20: Update keys $e(s)$ in *OPEN* and prune if $g(s) + h(s) \geq G$

Figure 1: The Anytime Nonparametric A* algorithm.

inserted into the *OPEN* queue with key $e(s')$, or if it was al-

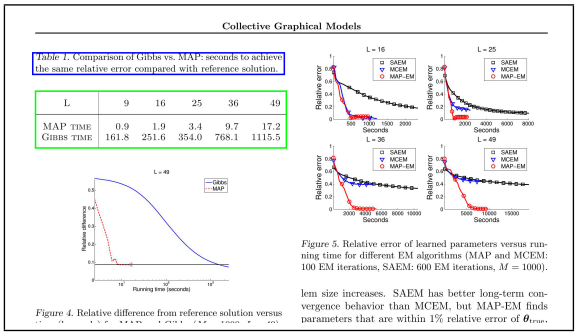
3.2 Suboptimality Bound

Our algorithm ANA* provides a suboptimality bound of the current-best solution that gradually decreases as the algorithm progresses. Each time a state s is selected for expansion in line 2 of IMPROVESOLUTION, its $e(s)$-value bounds the suboptimality of the current-best solution. We prove the theorem below. We denote by G^* the cost of the optimal solution, so G/G^* is the true suboptimality of the current-best solution (recall that G is the cost of the current solution). Further, we denote by $g^*(s) = c^*(s_{\text{start}}, s)$ the cost of the optimal path between the start state s_{start} and s.

Lemma: If the optimal solution has not yet been found, there must be a state $s \in OPEN$ that is part of the optimal solution and whose g-value is optimal, i.e. $g(s) = g^*(s)$.

Proof: Initially, $s_{\text{start}} \in OPEN$ is part of the optimal solution and $g(s_{\text{start}}) = g^*(s_{\text{start}}) = 0$. At each iteration a state s from *OPEN* is expanded. s is either part of the optimal solution and $g(s)$ is optimal, or not. In the latter case, the state with the above property remains in *OPEN* (its g-value is optimal and cannot be decreased). In the former case, s must have a successor s' that is part of the optimal solution. The successor's updated g-value is optimal since edge (s, s') is part of the optimal solution and $g(s') = g(s) + c(s, s')$. This continues until the goal state is dequeued with optimal g-value

Correct extraction of a figure composed mostly of text [Berg et al., AAAI 2011]



Collective Graphical Models

Table 1. Comparison of Gibbs vs. MAP: seconds to achieve the same relative error compared with reference solution.

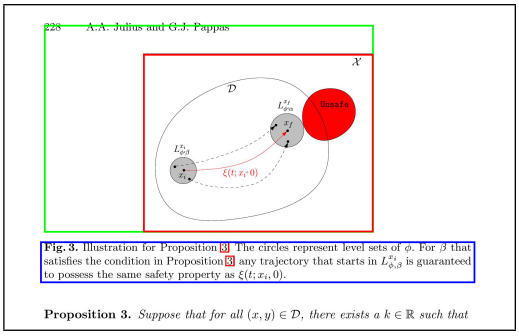
L	9	16	25	36	49
MAP TIME	0.9	1.9	3.4	9.7	17.2
GIBBS TIME	161.8	251.6	354.0	768.1	1115.5

Figure 4. Relative difference from reference solution versus

Figure 5. Relative error of learned parameters versus running time for different EM algorithms (MAP and MCEM: 100 EM iterations, SAEM: 600 EM iterations, $M = 1000$).

lem size increases. SAEM has better long-term convergence behavior than MCEM, but MAP-EM finds parameters that are within 1% relative error of θ_{true}.

Correct extraction of a table adjacent to a figure [Sheldon et al., ICML 2013]

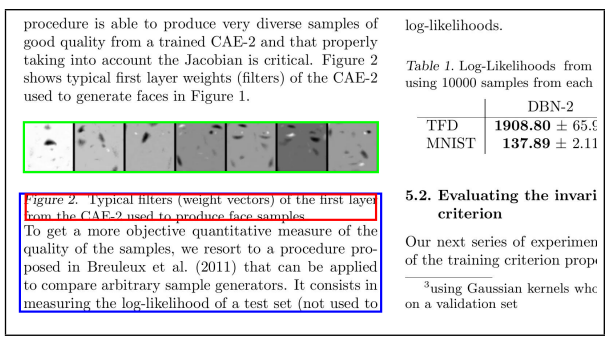


228 A.A. Julius and G.J. Pappas

Fig. 3. Illustration for Proposition 3. The circles represent level sets of ϕ. For β that satisfies the condition in Proposition 3, any trajectory that starts in $L^{\pi_I}_{\phi,\beta}$ is guaranteed to possess the same safety property as $\xi(t; x_I, 0)$.

Proposition 3. *Suppose that for all $(x, y) \in \mathcal{D}$, there exists a $k \in \mathbb{R}$ such that*

Error caused by mistaking a page header as part of the figure [Julius et al., HSCC 2009]



procedure is able to produce very diverse samples of good quality from a trained CAE-2 and that properly taking into account the Jacobian is critical. Figure 2 shows typical first layer weights (filters) of the CAE-2 used to generate faces in Figure 1.

Figure 2. Typical filters (weight vectors) of the first layer from the CAE-2 used to produce face samples.

To get a more objective quantitative measure of the quality of the samples, we resort to a procedure proposed in Breuleux et al. (2011) that can be applied to compare arbitrary sample generators. It consists in measuring the log-likelihood of a test set (not used to

log-likelihoods.

Table 1. Log-Likelihoods from using 10000 samples from each

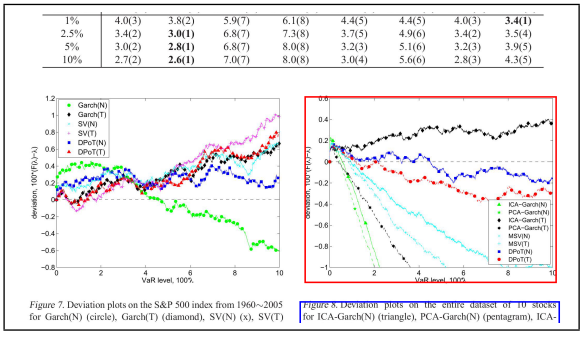
	DBN-2
TFD	**1908.80** ± 65.9
MNIST	**137.89** ± 2.11

5.2. Evaluating the invari criterion

Our next series of experimen of the training criterion prop

[3]using Gaussian kernels whc on a validation set

Error caused by including extra text in a caption [Rifai et al., ICML 2012]

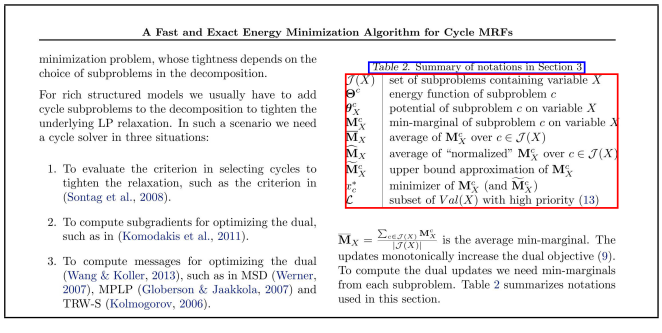


1%	4.0(3)	3.8(2)	5.9(7)	6.1(8)	4.4(5)	4.4(5)	4.0(3)	**3.4(1)**
2.5%	3.4(2)	**3.0(1)**	6.8(7)	7.3(8)	3.7(5)	4.9(6)	3.4(2)	3.5(4)
5%	3.0(2)	**2.8(1)**	6.8(7)	8.0(8)	3.2(3)	5.1(6)	3.2(3)	3.9(5)
10%	2.7(2)	**2.6(1)**	7.0(7)	8.0(8)	3.0(4)	5.6(6)	2.8(3)	4.3(5)

Figure 7. Deviation plots on the S&P 500 index from 1960∼2005 for Garch(N) (circle), Garch(T) (diamond), SV(N) (x), SV(T)

Figure 8. Deviation plots on the entire dataset of 10 stocks for ICA-Garch(N) (triangle), PCA-Garch(N) (pentagram), ICA-

Error caused by inability to separate a 3-way figure layout [Chen et al., ICML 2010]



A Fast and Exact Energy Minimization Algorithm for Cycle MRFs

minimization problem, whose tightness depends on the choice of subproblems in the decomposition.

For rich structured models we usually have to add cycle subproblems to the decomposition to tighten the underlying LP relaxation. In such a scenario we need a cycle solver in three situations:

1. To evaluate the criterion in selecting cycles to tighten the relaxation, such as the criterion in (Sontag et al., 2008).
2. To compute subgradients for optimizing the dual, such as in (Komodakis et al., 2011).
3. To compute messages for optimizing the dual (Wang & Koller, 2013), such as in MSD (Werner, 2007), MPLP (Globerson & Jaakkola, 2007) and TRW-S (Kolmogorov, 2006).

Table 2. Summary of notations in Section 3.

$\mathcal{J}(X)$	set of subproblems containing variable X
Θ^c	energy function of subproblem c
θ^c_X	potential of subproblem c on variable X
$\mathbf{M}^c_X$	min-marginal of subproblem c on variable X
$\overline{\mathbf{M}}_X$	average of $\mathbf{M}^c_X$ over $c \in \mathcal{J}(X)$
$\widetilde{\mathbf{M}}_X$	average of "normalized" $\mathbf{M}^c_X$ over $c \in \mathcal{J}(X)$
$\overline{\mathbf{M}}^c_X$	upper bound approximation of $\mathbf{M}^c_X$
c^*_X	minimizer of $\mathbf{M}^c_X$ (and $\widetilde{\mathbf{M}}_X$)
$\mathcal{L}$	subset of $Val(X)$ with high priority (13)

$\overline{\mathbf{M}}_X = \frac{\sum_{c \in \mathcal{J}(X)} \mathbf{M}^c_X}{|\mathcal{J}(X)|}$ is the average min-marginal. The updates monotonically increase the dual objective (9). To compute the dual updates we need min-marginals from each subproblem. Table 2 summarizes notations used in this section.

Error caused by mistaking table text as body text [Wang et al., ICML 2013]

Figure 4: Qualitative results. Figures that our system extracted are shown in green, captions are shown in blue, and, in the case of errors, correct extractions are marked in red. The top two rows show correct extractions, the third row shows incorrect extractions, and the last row shows figures our system failed to extract.

Dataset	Extractor	Locate	Render
CS-150	PDFFigures 2.0 (Ours)	0.21	2.52
	PDFFigures [5]	0.84	0.93
CS-Large	PDFFigures 2.0 (Ours)	0.27	1.84
	PDFFigures [5]	1.19	1.23

Table 3: Mean number of seconds required to locate, or to both locate and render, the figures in a paper. Our approach is considerably faster at locating figures, but slower when required to render the extracted regions as images.

Dataset	Source of Errors	Count	Percent
CS-150	Text Extraction	10	0.46
	Caption Detection	2	0.09
	Caption Extraction	2	0.09
	Text Classification	2	0.09
	Figure Assignment	2	0.09
	Cropping Error	2	0.09
	Other	2	0.09
CS-Large	Text Extraction	32	0.29
	Caption Detection	6	0.06
	Caption Extraction	15	0.14
	Text Classification	29	0.26
	Figure Assignment	14	0.13
	Cropping Error	12	0.11
	Other	1	0.01

Table 4: Error analysis of our approach. Inaccuracies when extracting text was a major source of error, followed by errors in text classification.

Figure 4 shows qualitative results from our method. PDFFigures 2.0 produces very clean extractions in most cases. Errors tend to emerge in papers where figures are arranged in complex layouts, or are caused by difficulty correctly classifying blocks of text that do not match the usual flow of body text, such as equations, page headers, or text inside figures.

6.2 Runtime Analytics

We measure our method's runtime performance on both datasets and compare to [5], as shown in Table 3. We measure the time it takes to return the location of the figures in a document, and also the time it takes to both locate the figures and save them as separate image files[2]. PDFFigures 2.0 is considerably faster at locating figure regions. However, rendering the figures using PDFBox [1] proved to be slower then rendering them using Poppler [2], the PDF rendering engine used by [5]. In the future, we might consider using a different library to complete the rendering step in order to remove this performance gap.

6.3 Error Analysis

We perform an error analysis to assess the performance of individual steps of our approach. The analysis is listed in Table 4. We categorize errors (both false positives and false negatives) into one of following six categories. Text extraction errors, referring to errors caused by text not being extracted correctly from the PDF. Caption detection errors, caused by failing to locate captions. Caption extraction errors, where a caption was correctly located but had incorrect text. Text classification errors, due to incorrectly classifying body text as figure text or vice versa. Figure assignment errors, caused by generating or selecting incorrect proposal regions during the figure assignment step. Or cropping errors, meaning the returned figure region was approximately correct, but had not been clipped in quite the same manner as the ground truth region. Cropping errors were often due to minor errors in the bounding boxes extracted for text or graphical elements.

Text extraction was a signfiicant source of error, some PDFs that appear correct in PDF viewers yield erroneous text when parsed programmatically. In many cases, other PDF parsing tools beside PDFBox [1], such as Poppler [2], also failed to parse these PDFs, implying these errors stem from how the text was encoded and would be hard to avoid. The next largest source of error was region classification, many errors caused by misclassifying text in page headers, bullet points, equations, or text inside text heavy figures.

7. SECTION TITLE EXTRACTION

While our focus is on extracting figures, the document decomposition produced by our method and our techniques for detecting anomalous text can be valuable when extracting other important elements from documents as well. In this section, we demonstrate how our approach can be extended for extracting section titles. A particular motivation for this task is that section titles can provide additional context when interpreting figures. We have also found open source tools like ParsCit [6] and Grobid [8] to be less effective for this task on our dataset.

7.1 Proposed Approach

Our approach detects lines of text that (I): Are either all caps, or have a font different than the most common font in the document, or are larger than the average font size in the document. (II): Have a uniform font and font size. (III): Have a larger than average amount of space between itself and the line above it. (IV): Have at least one line of text below it. (V): Are centered or left aligned to a column. (VI): Start with a number or are upper case. We remove lines that appear to be part of an equation by removing lines containing many non-alphabetic characters, and we remove lines that appear to be part of a list (ex. "Theorem 1:") by removing lines that end in a number. If any of the remaining lines are consecutively ordered, we have to determine whether they are part of a single, multi-line title or if the following line(s) are a separate section title. Similar to our method of locating captions (Section 4.1), we resolve this problem using the line's justification. If the font and justification of the second line matches the first line the lines are labelled as a multi-line section title, otherwise they are labelled as separate section titles. This phase is run only after the figures and tables have been extracted, since figures and tables often have their own titles which may produce false positives.

7.2 Results

An evaluation of our section title extraction approach was completed on 65 documents, 26 sampled from CS-150 and 39 sampled from CS-Large. For comparison we use the well known PDF parsing program Grobid [8]. We experimented

[2]Experiments were run on a single thread on a Macintosh OS X 10.10 with a 2.5GHz Intel core i7 processor.

Extractor	Precision	Recall	F1
PDFFigures 2.0 (Ours)	0.946	0.975	0.960
Grobid	0.701	0.818	0.755
Grobid, Numbered Only	0.934	0.670	0.781

Table 5: Precision-recall scores of Grobid and PDFFigures 2.0 for section title extraction.

with using all the sections extracted by Grobid, or just the sections that began with a number. We additionally filter out section titles produced by Grobid that were of length one or contained no alphabetic characters. The output of both algorithms was compared against manually extracted section titles. Our results are shown in Table 5. PDFFigures 2.0 shows considerably better performance on this dataset. Many of Grobid's errors are due to extracting bold lines of text that are not section titles, chunking section titles with the line beneath them, or extracting text within a figure or table as a section title. If we only use numbered sections, Grobid's precision becomes much better, although chunking errors still reduce precision while recall drops since many PDFs in our sample do not number their sections. False positives are the primary source of error in our algorithm and are often due to extracting bold lines of text that begin new paragraphs, but were not annotated as section titles.

8. INTEGRATING PDFFIGURES 2.0 INTO A DIGITAL LIBRARY

PDFFigures 2.0 has been featured in a smart online search engine for scholarly computer science documents, Semantic Scholar (www.semanticscholar.org). Semantic Scholar uses state of the art techniques in natural language processing to add semantics to scientific literature search. Integration with a complex, distributed document processing system such as Semantic Scholar requires that our approach is both scalable and easy to integrate into existing codebases. To achieve this goal, PDFFigures 2.0 was implemented in Scala so that it can be easily integrated into JVM based distributed processing environments. We also ensure that PDFFigures 2.0 can be timed out and interrupted if it stalls when parsing a PDF. Using the Apache Spark distributed framework [16], the extractor was run on over one million PDFs. We were able to mine figures from about 96% of these PDFs without errors. In total, 5 million figures and 1.4 million tables were extracted.

The extracted figures are used as part of the user interface of Semantic Scholar. Semantic Scholar features a summary page for each paper, which includes content such as the paper abstract, key phrases, the citations to and from the paper along with highlighted key citations. The extracted figures are shown to the user as thumbnails beneath the abstract. By clicking on an individual figure, users can view it at full scale, or flip through the other figures in the paper. Extracted captions are shown below each figure to provide further context. Figure 5 provides a snapshot of the Semantic Scholar UI. We expect the ability to preview figures in this manner to be especially helpful for mobile users. While it is normally very difficult to view figures on a mobile device, the Semantic Scholar mobile site presents users with a list of the figures in each paper that can be tapped on to be viewed in full screen.

Statistic	Correlation with Citations
Number of Figures	0.0332
Number of Tables	0.0634
Number of Figures and Tables	0.0535
Mean Figure Caption Length	0.0471
Mean Table Caption Length	0.0617
Mean Caption Length	0.0627

Table 6: Spearman rank correlation between figure usage and citations normalized by year and venue. We can observe a slight correspondence between using figures and including longer captions with citations. All values have two sided p-values with $p < 10^{-10}$.

9. EXPLORING FIGURE & TABLE USAGE

As a result of building a database of over 5 million figures (Section 8) we acquired a large collection of rich figure metadata. This metadata gave us an opportunity to investigate figure usage in computer science literature. In this section, we present some of the insights we were able to derive. Note that our analysis is limited to open source documents.

9.1 Figure Usage Over Time

We investigated how the usage of figures has changed over time. A plot of the mean number of figures and tables used in papers during different years can be found in Figure 6. We found both figure and table usage has increased over time, with figure usage undergoing a specially large increase during the years of 1995-2005. The increase in the usage of figures over time provides some empirical evidence that figures have become an increasingly important part of the computer science literature. We also examine how the length of captions has changed over time. The plot in Figure 7 shows that captions tend to be longer in more recent papers, which also suggests authors have devoted more space to presenting and explaining figures in recent years.

9.2 Figures and Citations

We examined how figure usage correlates with the number of citations a paper receives. To control for factors such as time of publication, page count, general topic, and conference prestige, we normalize each paper's citations by conference. That is, we subtract from each paper's citation count the mean number of citations given to papers that were published in the same conference and year. To ensure accurate normalization, we only make use of a paper if there were 30 other papers from the same conference and year. We then compute the Spearman rank correlation between number of figures used and mean caption length with citations. Our results are shown in Table 6. We found that both number of tables and number of figures has a small correlation with the number of citations a paper receives, and that this trend is stronger for tables than it is for figures. Caption length also has a correlation with citations, again with a stronger effect towards table caption length. Although these values are small, they are statistically significant and are sufficient to indicate a link between citations and figure usage. One possible explanation is that papers with extensive empirical results, which often leads to higher figure usage, tend to get cited more frequently. Another plausible explanation for this correlation is that leading researchers and authors tend

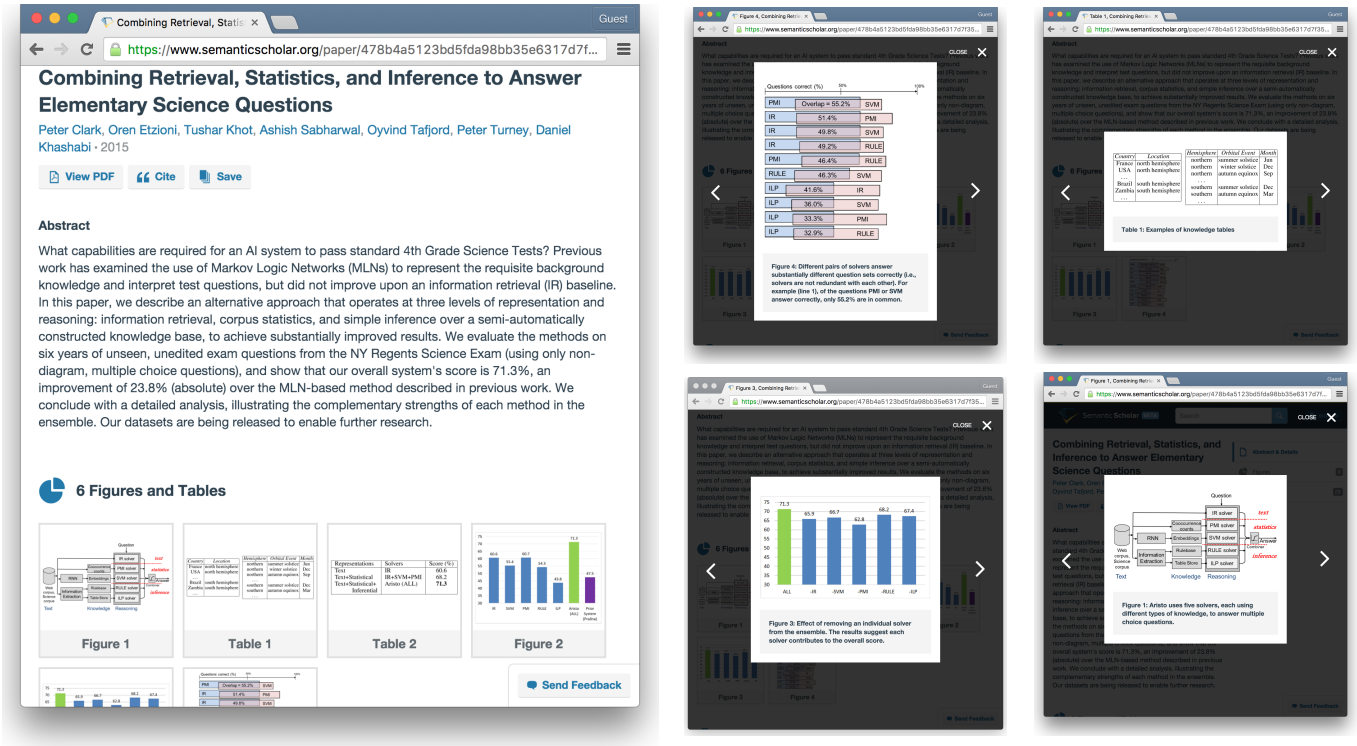


Figure 5: User interface with the extracted figures and tables from Semantic Scholar. Figures are shown as thumbnails below the abstract (left). Users can select a figure to see the full image and its caption (right). This real world example also demonstrates PDFFigures 2.0's ability to extract many different kinds of figures

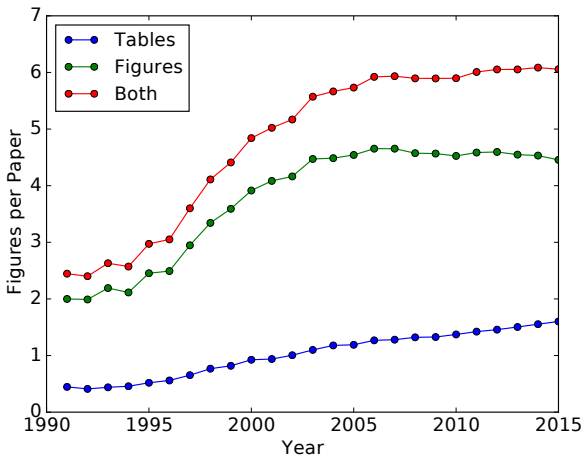


Figure 6: Mean number of tables and figures in papers published in different years. Papers have used more tables and figures in recent years.

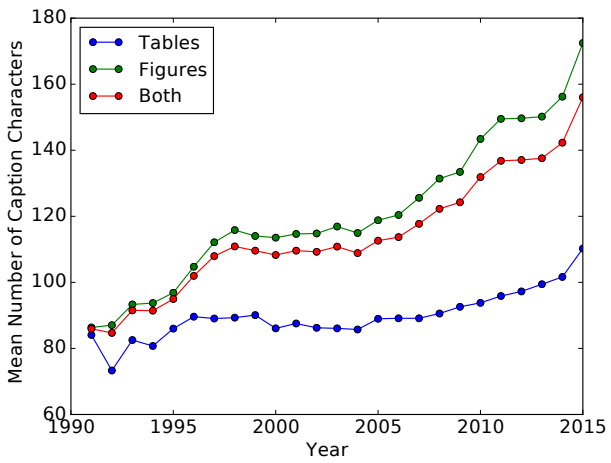


Figure 7: Mean caption length in papers published in different years. Typical caption length has increased over time.

Conference	Tables Per Paper
Empirical Methods on Natural Language Processing (EMNLP)	3.86
Conference of Evaluation Information Access Technologies (NTCIR)	3.77
International Joint Conference on Natural Language Processing (IJCNLP)	3.50
Text Analysis Conference (TAC)	3.46
Automatic Speech Recognition and Understanding Workshop (ASRU)	3.44

Table 7: Conferences with the most tables in each paper. This category is dominated by natural language processing conferences.

Conference	Figures Per Paper
International Meshing Roundtable (IMR)	10.23
International Conference on Mobile Systems (MOBISYS)	9.39
Symposium on High-Performance Computer Architecture (HPCA)	8.92
Internet Measurement Conference (IMC)	8.90
European Conference on Computer Systems (EuroSys)	8.06

Table 8: Conferences with the most figures in each paper. Many system conferences can be observed.

to both use more figures and accrue more future citations.

9.3 Figures by Conferences

Finally, we investigate how figure and table usage varies between conferences. We compute the mean number of figures and tables used by papers in different conference venues, only including venues for which we had at least 150 documents. The venues with the highest table usage can be found in Table 7, and the venues with the highest figure usage in Table 8. In our dataset, we found that natural language processing conferences dominate the top spots for using tables, an indicator that empirical results and making comparisons has been especially important in that field. The conferences where figures were most frequently used include many system conferences, which tend to both have higher page limits and make frequent use of figures to illustrate circuit or hardware layouts.

10. CONCLUSION

In this paper we considered the challenging problem of developing a scalable figure extraction method that is robust enough to be used across the entire range of content in a digital library. Our contributions include a set of widely applicable text classification heuristics, a clustering mechanism for detecting figure regions, and a novel section title extraction method. Evaluation on manually annotated real world documents and integration with the Semantic Scholar search engine shows the success of our approach. Extracting data from figures and making use of them in user interfaces is an exciting new line of research in digital libraries. We hope that PDFFigures 2.0 will expand upon PDFFigure's success facilitating research in these new fields. While our approach achieves very high accuracy in the computer science literature, future work includes adapting our method to be effective in additional scholarly domains. Combining our heuristic approach with machine learning based approaches is also an interesting avenue for future work.

11. ACKNOWLEDGMENTS

We thank Isaac Cowhey for his help in annotating our dataset. We also thank the Semantic Scholar team for helping with the integration of PDFFigures 2.0.

12. REFERENCES

[1] PDFBox. https://pdfbox.apache.org/.
[2] Poppler. https://poppler.freedesktop.org/.
[3] Semantic Scholar. www.semanticscholar.org.
[4] Text Detection in Screen Images with a Convolutional Neural Network. https://github.com/domoritz/label_generator.
[5] C. Clark and S. Divvala. Looking Beyond Text: Extracting Figures, Tables, and Captions from Computer Science Paper. In *AAAI, Workshop on Scholarly Big Data*, 2015.
[6] I. G. Councill, C. L. Giles, and M.-Y. Kan. ParsCit: An Open-source CRF Reference String Parsing Package. In *LREC*, 2008.
[7] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. In *IJCV*, 2010.
[8] P. Lopez. GROBID: Combining Automatic Bibliographic Data Recognition and Term Extraction for Scholarship Publications. In *Research and Advanced Technology for Digital Libraries*, 2009.
[9] M.-T. Luong, T. D. Nguyen, and M.-Y. Kan. Logical Structure Recovery in Scholarly Articles with Rich Document Features. In *IJDLS*, 2011.
[10] P. A. Praczyk and J. Nogueras-Iso. Automatic Extraction of Figures from Scientific Publications in High-Energy Physics. In *Information Technology and Libraries*, 2013.
[11] S. Ray Choudhury, P. Mitra, and C. L. Giles. Automatic Extraction of Figures from Scholarly Documents. In *DocEng*, 2015.
[12] P. M. Sagnik Choudhury, Shuting Wang and L. Giles. Automated Data Extraction from Scholarly Line Graphs. In *GREC*, 2015.
[13] D. Sculley, T. Phillips, D. Ebner, V. Chaudhary, and M. Young. Machine learning: The high-interest credit card of technical debt. In *NIPS Software Engineering for Machine Learning Workshop*, 2014.
[14] N. Siegel. Understanding Charts in Research Papers: A Learning Approach. Technical report, University of Washington, 2015.
[15] J. Wu, J. Killian, H. Yang, K. Williams, S. R. Choudhury, S. Tuarob, C. Caragea, and C. L. Giles. PDFMEF: A Multi-Entity Knowledge Extraction Framework for Scholarly Documents and Semantic Search. In *K-CAP*, 2015.
[16] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster Computing with Working Sets. In *USENIX Conference on Hot Topics in Cloud Computing*, 2010.