Evaluating Document Analysis Results via Graph Probing

Daniel Lopresti and Gordon Wilfong
Bell Labs, Lucent Technologies Inc.
600 Mountain Avenue
Murray Hill, NJ 07974 USA
{dpl,gtw}@research.bell-labs.com

Abstract

While techniques for evaluating the performance of lower-level document analysis tasks such as optical character recognition have gained acceptance in the field, attempts to formalize the problem for higher-level algorithms that incorporate more complex structure have been less successful. In this paper, we describe an intuitive, easy-to-implement scheme for the problem of performance evaluation when document recognition results are represented in the form of a directed acyclic graph. We present results from two simulation studies based on different graph models and one experiment using a well known page segmentation algorithm to demonstrate the applicability of the approach.

1 Introduction

As document analysis systems grow more sophisticated, it becomes increasingly important to be able to evaluate their performance. With a few notable exceptions, however, little has been achieved along these lines beyond the informal assertions that often accompany work published in the field.

The directed acyclic graph, or *DAG*, is nearly universal across recognition algorithms, but this rich representation is typically discarded when it comes time for evaluation. Although there is already a substantial amount of theory for the problem of evaluating logical structure recognition, the empirical literature has largely ignored this work and usually resorts to a manual approach to evaluation: counting by hand the number of components that have been missed or added.

In this paper, we examine in detail a paradigm we first put forth in the context of our work on table recognition [1, 2]. This methodology, known as "graph probing," offers an intuitive, easy-to-implement scheme for the general problem of evaluating document recognition when the results are represented in the form of a DAG.

2 Graph Probing

Given the DAG for a recognition result and for its corresponding ground-truth, it is natural to consider comparing the two as a way of determining how well an algorithm has done. Attempting this directly, however, gives rise to two dilemmas. The first is that any reasonable notion of graph matching subsumes the graph isomorphism problem, for which no efficient algorithm is known to exist. The other obstacle is that there may be several different ways to represent the same logical structure as a graph, all equally applicable. Minor discrepancies could create the appearance that two graphs are dissimilar when in fact they are functionally equivalent from the standpoint of the intended application.

At the other end of the spectrum, we could embed the recognition algorithm in a complete, end-to-end system and measure the system's performance on a specific task from the user's perspective: Does it provide the desired information? This approach has its own shortcomings, however, as it limits the generality of the results and makes it difficult to identify the precise source of errors that arise when complex processes interact.

We have developed a third methodology that lies midway between these two. We work directly with the graph representation. However, instead of trying to match the graphs under a formal model, we probe their structure and content by asking relatively simple queries that mimic, perhaps, the sorts of operations that might arise in a real application.

Conceptually, the idea is to place each of the two graphs under study inside a "black box" capable of evaluating a set of graph-oriented operations (*e.g.*, returning a list of all the leaf nodes, or all nodes labeled in a certain way). We then pose a series of probes and correlate the responses of the two systems. A measure of their similarity is the number of times their outputs agree, as depicted in Figure 1.

As was observed earlier, the problem studied here is clearly related to the problem of graph isomorphism, the

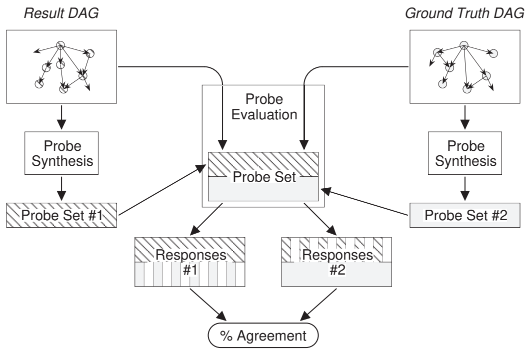

Figure 1. Overview of graph probing.

complexity of which remains open. Many heuristics for determining isomorphism have relied on using *vertex invariants*, where a vertex invariant consists of a value $f(v)$ assigned to each vertex v, so that under any isomorphism I, if $I(v) = v'$ then $f(v) = f(v')$. One such vertex invariant is the degree of the vertex (or the in- and out-degrees, if the graph is directed). In fact **nauty**, a successful software package for determining graph isomorphism (see [4]), relies on such vertex invariants.

This type of result motivates the idea of performing local probes to try to determine if there exist differences in a pair of graphs. However, we wish to solve more than this simple “yes/no” problem; we are also interested in quantifying the similarity between two graphs. While the probing paradigm is open-ended, currently we have defined three categories of probes that are applicable across all of the graphs in our studies:

Class 0 These probes verify the occurrence of a given type of node in the graph. A possible Class 0 probe might be paraphrased as: *Does the graph contain a node labeled “Line”?*

Class 1 These probes combine content and label specifications. A typical probe might be: *Does the graph have a node labeled “Word” with content “pentagon”?*

Class 2 These probes examine the node and edge structure of the graph by checking in- and out-degrees. An example is: *Does the graph have a node with in-degree 2 and out-degree 2?*

Additional power can be obtained by allowing the graph to be marked-up in response to probing. For example, we can confirm that a graph has at least n nodes labeled “Line” by repeating the following probe n times: *Does the graph contain an unmarked node labeled “Line”? If so, mark it.*

The generation of a probe set is based on one or the other of the graphs in question (recall Figure 1). That graph will obviously return the definitive responses for all of the probes in the set, while the other graph will do more or less well depending on how closely it matches the first. The process is then repeated from the other direction, generating the probe set from the second graph and tallying the responses for both. The probes are synthesized automatically, working from the graphs output by the recognition and ground-truthing processes.

We define a *discriminating probe* to be a probe that demonstrates a difference between two graphs. Two fundamental questions are of interest: (1) For two graphs that are different, does there exist at least one discriminating probe? and (2) Over the entire set of probes, how many are discriminating? The first of these reflects the graph isomorphism problem. The second can serve as a measure of how similar the two graphs are. To make this more explicit, we define the *agreement* between two probe sets to be:

$$\textit{agreement} \equiv 1.0 - \frac{\textit{\# of discriminating probes}}{\textit{total \# of probes}} \tag{1}$$

Our aim is to equate “agreement” with the traditional concept of “accuracy.”

Several criteria are desirable when designing a probing strategy: probes should be invariant across graph isomorphism, they should be easy to evaluate and the responses easy to compare, similar graphs should agree more often than dissimilar graphs, and the probes in a set should be independent.

An approach that is philosophically quite similar to ours compares graphs based on their vertex degree histograms [6]. We employ a range of invariants, however, while that work uses only the one.

3 Experimental Results

To test the concept of graph probing, we designed a series of simulation studies as well as an experiment using results from a well known page segmentation algorithm. As indicated, we would like to be able to equate probing agreement (*i.e.*, Equation 1) with some general notion of accuracy. Unfortunately, there is no measure that is both universal and easy-to-compute which we can use for comparison purposes (indeed, this point is a primary motivation of our research). Hence, we have chosen to work “backwards” by randomly generating a ground-truth graph, and then simulating recognition “errors” by editing the graph in various ways: adding and deleting nodes, altering labels and content, etc. The number of edits we perform is an approximation (an upper bound, in fact) of the true distance between two DAG's.

3.1 Entity Graph Model Simulation

The *entity graph* model reflects a standard document hierarchy: nodes labeled as *Page*, *Zone*, *Line*, or *Word*. The edge structure represents two relationships: *contains* and *next*.

Our procedure begins by creating a graph for a page with a random number of zones. For each zone, we then generate a random number of lines, and for each line a random number of words. Content for *Word* nodes is chosen to be either: (1) a word randomly selected from the Unix **spell** dictionary, or (2) a random integer. The editing operations used to simulate recognition errors are guaranteed to yield another legal entity graph. These include altering the content of a *Word* node, deleting an existing *Word*, *Line*, or *Zone* node (and its associated edges), or inserting a new *Word*, *Line*, or *Zone* node.

The entire simulation involved generating 500 "ground-truth" entity graphs, performing a randomly selected number of edits on each, synthesizing and evaluating Class 0, 1, and 2 probes, and gathering relevant statistics. The study required about 4.5 hours to run.

The results for the entity graph experiment are presented in Table 1 and Figure 2. As can be seen from the upper table, there was a wide range in the size of the graphs under consideration. On average, approximately three probes were generated per node, and each pair of graphs required about half a minute to compare via probing. Overall, the average probing agreement was 0.911, and the maximum was 0.999 (*i.e.*, the probes always captured the fact that one of the graphs contained errors).

Attribute	*Min*	*Max*	*Ave*
Zones	1	8	4.6
Lines	1	51	20.8
Words	2	270	103.7
Total Nodes	5	330	130.2
Edits	1	63	14.9
Class 0 Probes	19	653	260.3
Class 0 Agreement	0.172	1.000	0.926
Class 1 Probes	12	546	208.6
Class 1 Agreement	0.093	0.998	0.897
Class 2 Probes	19	653	260.3
Class 2 Agreement	0.138	1.000	0.905
Overall Probes	50	1,852	729.2
Overall Agreement	0.138	0.999	0.911
Probes/Node	2.632	2.877	2.792
Probe Time (secs)	0.790	120.670	26.015
Secs/Probe	0.012	0.065	0.029

Probes	*Detected*	*Missed*	*Rate*	*Unique*
Class 0	455	45	91.0%	0
Class 1	500	0	100.0%	34
Class 2	466	34	93.2%	0
Overall	500	0	100.0%	n/a

Table 1. Statistics for the entity graph simulation.

The ability of the three probe classes to differentiate the two graphs is shown in the lower part of Table 1. Class 1 probes never failed in this experiment. Note that Class 0 and Class 2 probes will always miss differences that involve only content, but off-setting edits have the potential to confuse any of the classes. The last column in this table indicates that there were 34 graph-pairs that were distinguished only by using Class 1 probes.

The number of discriminating probes as a function of graph editing operations is displayed in Figure 2. The datapoints show the average at each step along the x-axis, while the vertical bars give the min/max range. Turning this around, the size of the discriminating probe set provides a reasonably dependable measure of the difference between two graphs. It is likely that refining and/or weighting the probe sets appropriately could lead to an improvement in the "outliers," a topic for future research.

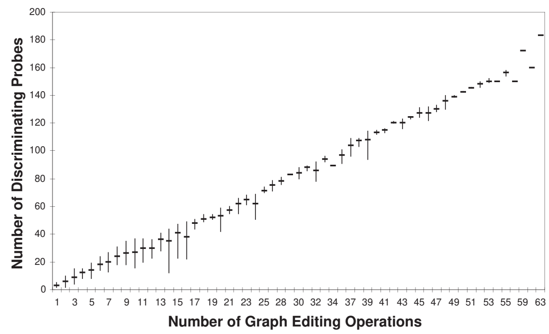

Figure 2. Results for the entity graph simulation.

3.2 Table Graph Model Simulation

Entity graphs encode document page structure in a very general way. A more restricted type of graph is the *table graph*, as defined in our past work on table recognition [1, 2]. Tables consist of lower-level cells, grouped in terms of logical rows and columns. Hence, nodes in table graphs can be labeled *Cell*, *Row*, and *Column*. Edges encode the *contains* relationship. For the table graph model, we add a fourth, more sophisticated class of probes:

Class 3 For a given target node, keys that uniquely determine its row and column are identified. These are used to index into the graph, retrieving the content of the node that lies at their intersection. An example is: *What is the content of the cell that lies at the intersection of the row indexed by "Overall Agreement" and the column indexed by "Max"?*

Along the same lines as the previous simulation, we begin by generating a ground-truth graph containing a random number of rows and columns. Each column is randomly designated as being either alphabetic or numeric. For the former, table cells are selected to be a string of one or more words, while for the latter the contents of cells are assigned to be random integers. Cells in the first row and column are always set to be alphabetic (to represent table headers). Editing operations include changing the contents of a *Cell* node, or deleting or inserting a *Row* or *Column*.

Table 2 and Figure 3 present the results for running this simulation for 500 random tables. While these graphs were smaller than those for the entity graph experiment, the compute-time was somewhat higher owing to the new probe class. As Table 2 indicates, the Class 1 and 3 probes never failed. Overall, the average agreement was found to be 0.808.

Attribute	*Min*	*Max*	*Ave*
Rows	2	15	8.5
Cols	2	6	4.0
Total Nodes	8	111	46.8
Edits	1	22	9.3
Class 0 Probes	16	215	93.7
Class 0 Agreement	0.593	1.000	0.911
Class 1 Probes	8	210	76.3
Class 1 Agreement	0.000	0.988	0.838
Class 2 Probes	16	215	93.7
Class 2 Agreement	0.296	1.000	0.771
Class 3 Probes	6	174	67.7
Class 3 Agreement	0.000	0.974	0.662
Overall Probes	48	814	331.5
Overall Agreement	0.360	0.992	0.808
Probes/Node	3.000	3.850	3.460
Probe Time (secs)	0.840	162.860	30.951
Secs/Probe	0.016	0.207	0.067

Probes	*Detected*	*Missed*	*Rate*	*Unique*
Class 0	436	64	87.2%	0
Class 1	500	0	100.0%	0
Class 2	433	67	86.6%	0
Class 3	500	0	100.0%	0
Overall	500	0	100.0%	n/a

Table 2. Statistics for the table graph simulation.

The detection capabilities of the various probe classes are listed in the lower part of Table 2. The Class 0 and 2 probes exhibit more misses than they did in Table 1. Note that there was no instance where a class of probes found a difference that escaped all of the other classes.

The range in discriminating probes as a function of editing operations is charted in Figure 3. As before, the number of such probes appears to be a good predictor of the number of edits used to simulate recognition errors, although the behavior of graph-pairs near the extremes of the ranges merits closer examination.

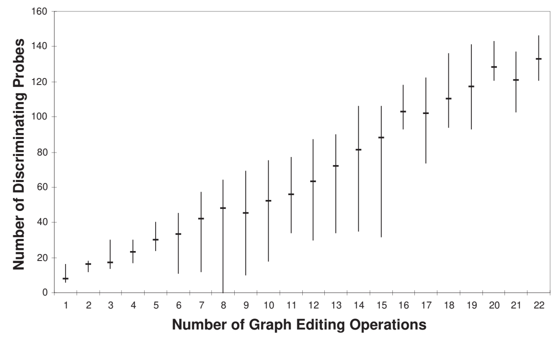

Figure 3. Results for the table graph simulation.

3.3 Page Segmentation Experiment

Our past experience using graph probing for performance evaluation in small-scale experiments involving real (as opposed to simulated) document analysis results has been quite favorable (see [1, 2]). As is often the case, however, the considerable effort required to create the necessary ground-truth presents a barrier to performing larger studies featuring our table understanding work at the present time (for a discussion of some of the associated issues, see [3] elsewhere in these proceedings). Instead, we developed a test using a well known page segmentation technique, Nagy and Seth's X-Y cut algorithm [5]. This partitions a page image recursively, representing the result as a tree. By injecting a controlled amount of random "bit-flip" noise (turning black pixels white as might arise in a light photocopy), we can induce differences in the segmentation graph that hopefully will be reflected in the agreement computed during probing.

The test collection consisted of 10 pages taken from the UW1 dataset: a037, c003, e011, e02j, h01c, h015, ig05, j002, n039, and s03n. We chose examples with relatively complex layouts, so that the X-Y cut graphs would be interesting. Each page was subjected to the bit-flip noise at rates ranging from 5% to 75% in increments of 5%. We then compared the segmentation graphs for the original and noisy pages using our graph probing paradigm.

Basic statistics for the probing evaluation of the 150 test pages are presented in Table 3. There were errors in every one of the recognized pages, and this is reflected in the maximum overall agreement which is 0.996. As in the simulations, a relatively small number of probes were generated for each node in the graphs. The probing time averaged 188 seconds per document page, which appears quite modest considering the sizes of the graphs involved.

The lower portion of Table 3 shows that, in nearly every case, all of the probe classes were capable of detecting that there were differences between the segmentation

Attribute	*Min*	*Max*	*Ave*
X-Cuts	299	1,096	510.9
Y-Cuts	59	659	210.4
Text	247	993	480.5
Other	1	29	10.3
Total Nodes	641	2,480	1,212.1
Class 0 Probes	1,284	4,759	2,424.2
Class 0 Agreement	0.776	1.000	0.959
Class 1 Probes	539	2,009	1,001.7
Class 1 Agreement	0.057	0.976	0.565
Class 2 Probes	1,284	4,759	2,424.2
Class 2 Agreement	0.779	1.000	0.960
Overall Probes	3,107	11,527	5,850.1
Overall Agreement	0.661	0.996	0.892
Probes/Node	2.384	2.443	2.412
Probe Time (secs)	48.270	690.110	188.529
Secs/Probe	0.016	0.060	0.028

Probes	*Detected*	*Missed*	*Rate*	*Unique*
Class 0	146	4	97.3%	0
Class 1	150	0	100.0%	4
Class 2	146	4	97.3%	0
Overall	150	0	100.0%	n/a

Table 3. Statistics for the page segmentation experiment.

graphs for the original and degraded documents. Only in four instances did the Class 1 probes outperform the other two.

The remaining issue, then, is how well graph probing correlates with the controlled amount of damage we inflicted on each page image. These results are plotted in Figure 4. Here we show a distinct style of datapoint for each of the 10 source documents in the test set. While the correspondence is perhaps "rougher" than in the simulations, an overall-monotonic behavior is still visible.

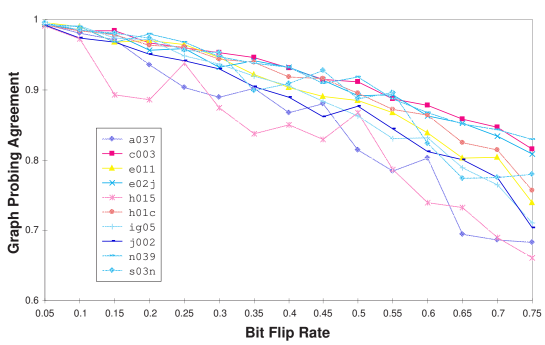

Figure 4. Results for the page segmentation experiment.

4 Conclusions

There are a number of ways in which this work could be extended. The design of optimal probe sets and/or weighting schemes is an open question. Beyond experimental studies, it should be possible to develop formal assertions about various classes of probes and their abilities to detect certain kinds of errors with high probability. The probing paradigm as we have defined it is an off-line procedure (*i.e.*, all of the probes are computed in advance). Allowing the probing to take place on-line, making it adaptive, might add significant power.

References

[1] J. Hu, R. Kashi, D. Lopresti, and G. Wilfong. A system for understanding and reformulating tables. In *Proceedings of the Fourth IAPR International Workshop on Document Analysis Systems*, pages 361–372, Rio de Janeiro, Brazil, December 2000.

[2] J. Hu, R. Kashi, D. Lopresti, and G. Wilfong. Table structure recognition and its evaluation. In *Proceedings of Document Recognition and Retrieval VIII*, volume 4307, pages 44–55, San Jose, CA, January 2001.

[3] J. Hu, R. Kashi, D. Lopresti, G. Wilfong, and G. Nagy. Why table ground-truthing is hard. In *Proceedings of the Sixth International Conference on Document Analysis and Recognition*, Seattle, WA, September 2001. To appear.

[4] B. McKay. Practical graph isomorphism. *Congressus Numerantium*, 30:45–87, 1981.

[5] G. Nagy and S. Seth. Hierarchical representation of optically scanned documents. In *Proceedings of the Seventh International Conference on Pattern Recognition*, pages 347–349, Montréal, Canada, July 1984.

[6] A. N. Papadopoulos and Y. Manolopoulos. Structure-based similarity search with graph histograms. In *Proceedings of the 10th International Workshop on Database & Expert Systems Applications*, pages 174–178. IEEE Computer Society Press, 1998.