Evaluating Document Analysis Results via Graph Probing*

Daniel Lopresti **Gordon Wilfong**
Bell Laboratories
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.

The paradigm, which we call "graph probing," has a sound basis in past work on heuristics for solving the graph isomorphism problem. However, our goal extends beyond simply testing for equivalence; we also wish to be able to quantify the similarity between two graphs. The technique described in this paper provides such a measure. We present results from three simulation studies based on different graph models and one experiment using real OCR data to demonstrate the applicability of the approach.

1 Introduction

As document analysis systems grow more and more sophisticated, it becomes increasingly important to be able to evaluate and compare 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. A thoughtful overview of the subject of automated performance evaluation can be found in [19].

While the directed acyclic graph, or *DAG*, is a nearly universal representation across recognition algorithms, the graph structure is typically discarded when it comes time for evaluation. In the case of page segmentation, for example, several practical approaches have been proposed based on distance-type measures, but these make little or no use of the whole graph, and instead focus on pixel-level comparisons [13, 26] or matching the text characters output from OCR [2, 13].

There is already a substantial amount of theory for the problem of evaluating logical structure recognition (see, e.g., [12, 18, 22–24]). Nevertheless, the empirical literature has largely ignored this work, perhaps owing to its complexity, and usually resorts to a simple, manual approach to evaluation: counting by hand the number of components that have been missed or added (e.g., [21]).

In this paper, we examine in detail a paradigm we first put forth in the context of our work on table recognition [10, 11]. 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, and may be extensible to other applications as well. Our approach uses a probing process to assess the agreement between the DAG returned by a recognition system and the DAG created during ground-truthing. Since different classes of probes are possible, ranging from very low- to very high-level, from concrete to abstract, this paradigm can be viewed as subsuming existing techniques that try to measure the structural similarity of the graph representations on the one hand, and the effectiveness of recognition results when incorporated in a particular application on the other.

We begin by describing the concept of graph probing in Section 2. After this overview, we present three different graph models in Section 3 that will be used in examining how well graph probing might work in practice. In Section 4, we discuss the results from three simulation studies and one experiment using real OCR data to demonstrate the applicability of the approach. Finally, we offer our conclusions and topics for future research in Section 5.

*Presented at the *Symposium on Document Image Understanding Technology*, Columbia, MD, April 2001.

2 Graph Probing

Given the DAG for a recognition result and the DAG 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, the complexity of which is open, as well as possibly the subgraph isomorphism problem, which is known to be NP-complete [9]. Hence, it seems unlikely that there exists an efficient, guaranteed-optimal algorithm for comparing two DAG's in the general case. While heuristics have been developed that are sometimes fast, their worst-case behavior is still exponential (see, e.g., [18]).

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. Forcing one graph to correspond to the other through a rigidly defined matching procedure obscures this important point.

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 is "goal-directed evaluation" as discussed in [19]). 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. This process is depicted in Fig. 1. Note that it is essential the probes themselves have simple answers that are easily compared. They might return, for example, a count of the number of nodes satisfying a certain property (e.g., possessing a particular label), or the content of a designated leaf node. The probing becomes recursive if the target of a probe is a graph itself. The intention is that this probing process abstracts the access of content away from the specific details of the graph's structural representation.

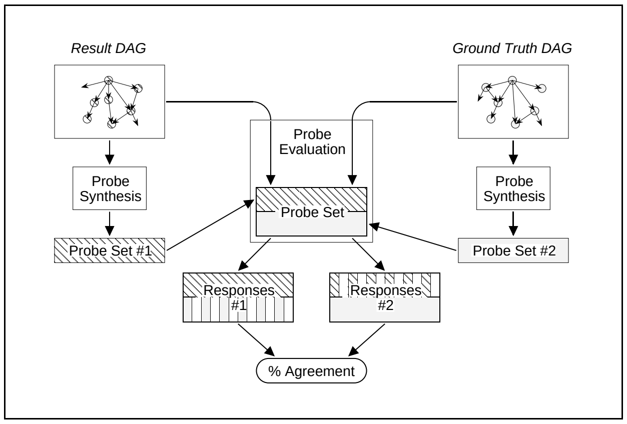

Figure 1: Overview of graph probing.

As noted earlier, the problem studied here is clearly related to the problem of graph isomorphism. The complexity of graph isomorphism remains open and, unfortunately, all known deterministic algorithms have worst-case exponential running times [8]. 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). There are numerous applications where graph isomorphism arises, such as mathematical chemistry [25], knowledge retrieval [6], robotics [5] and object recognition [1], and in these cases vertex invariants are often used to try to determine isomorphism. In fact, **nauty**, a successful software package for determining graph isomorphism (see [16, 17]), relies on vertex invariants.

There has been some analysis showing that such heuristics for determining graph isomorphism can fail in a catastrophic manner [4]. On the other hand, it has been proven that for random graphs, there is a very simple linear time test for checking if two graphs are isomorphic that is based on the degree of the nodes of the graphs, and this test succeeds with high probability [3]. This type of result motivates the idea of performing local probes to try to determine if there exist differences in a pair of graphs. In fact, these local probes will likely provide us with sufficient evidence to determine whether or not the two graphs are isomorphic. However, we wish to solve more than this simple "yes/no" problem; we are interested in *quantifying* the similarity between two different graphs.

Three specific directed acyclic graph models will be described in the next section. As is common in the literature, all incorporate nodes that are labeled as to type. In addition, some nodes also contain content. The set of types is relatively small and fixed in advanced, while content is unconstrained and open-ended. Edges, possibly labeled, express relationships between nodes.

While the probing paradigm is open-ended, currently we have defined three categories of probes for the graphs in our studies:

Class 0 These probes count the number of occurrences of a given type of node in the graph. A typical Class 0 probe might be paraphrased as: *How many nodes labeled "Line" does the graph have?*

Class 1 These probes combine content and label specifications. A representative Class 1 probe might be: *How many nodes labeled "Word" with content "pentagon" does the graph have?*

Class 2 These probes examine the node and edge structure of the graph by counting in- and out-degrees. An example of a Class 2 probe is: *How many nodes have in-degree 2 and out-degree 2?*

The generation of a probe set is based on one or the other of the graphs in question (recall Fig. 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. We then repeat the process 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 DAG's that are output by the recognition and ground-truthing processes. For specifying probes, we have implemented a graph-oriented query language embedded in a general-purpose programming language, Tcl/Tk [20]; this offers a great deal of flexibility.

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:

$$agreement \equiv 1.0 - \frac{\#\ of\ discriminating\ probes}{total\ \#\ of\ probes} \qquad (1)$$

If the agreement is 1.0, then the two graphs are indistinguishable with respect to the probe set in question. Values less than 1.0 indicate some degree of similarity falling short of a perfect match. Our aim is to equate "agreement" with the traditional concept of "accuracy."

3 Graph Models

In this section, we describe the three graph models we use in our experiments. As will be discussed later, we have written programs to randomly generate instances of graphs of a given type, as well as to edit graphs in ways reminiscent of recognition errors.

3.1 Entity Graph Model

The *entity graph* model reflects a standard document hierarchy: nodes labeled as *Page*, *Zone*, *Line*, or *Word* [14].[1] The edge structure represents two relationships: *contains* and *next*. An example of one such entity graph is shown in Fig. 2, corresponding to the nonsense document fragment given below:

```
satisfactory extrinsic inexpert frankfurter

abutting tarantula
grillwork pentagon attribution bilharziasis
```

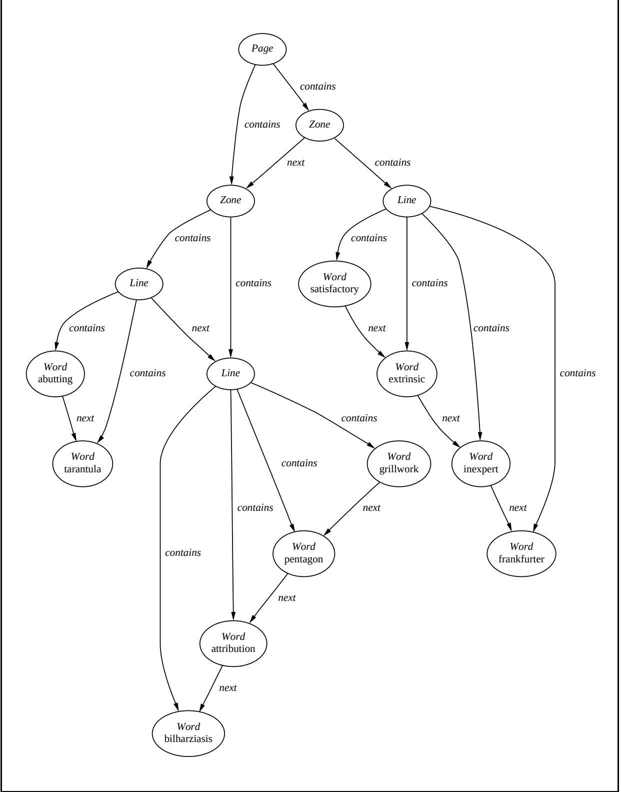

Figure 2: An instance of an entity graph.

[1] The entity model as it appears in [14] also includes *Char* as another level below *Word*. We ignore that refinement here for efficiency reasons.

3.2 Table Graph Model

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 [10, 11]. 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. An example of a table graph as derived from the following randomly generated table is shown in Fig. 3:

regression radiant	gusset prick	sima Nostrand
clubroom incubi	593134723	ant Sussex
ascribe gam	1813217419	opulent
shovel registrable	615003753	astride Peru

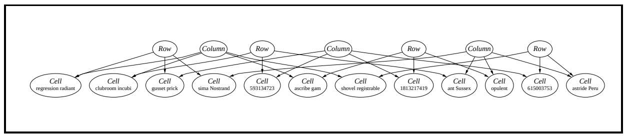

Figure 3: An instance of a table graph.

For the table graph model, we add a fourth, more sophisticated class of probes:

Class 3 These probes mimic simple database-style queries, although phrased entirely in terms of graph manipulations. 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 (if any) that lies at their intersection. An example of a Class 3 probe for the graph in Fig. 3 is: *What is the content of the cell that lies at the intersection of the row indexed by "ascribe gam" and the column indexed by "astride Peru"?* The response would be: *opulent*.

Class 3 probes are particularly interesting in that they lie at a higher level of abstraction than the other, simpler kinds of probes. Indeed, if the application in question was to build an interactive table look-up system, it could be argued that the results of Class 3 probing are more important than, say, counting the number of nodes labeled a certain way. Two graphs could be structurally quite different, but still respond similarly to Class 3 probes; to the user, they would be functionally the same.

3.3 Random Graph Model

The final model we consider in our experiments, the *random graph* model, is a completely random directed acyclic graph. As with the previous two models, nodes are labeled with a type and optional content. There is only one kind of edge, so these are not labeled. However, unlike the entity and table graphs, there are no restrictions on what constitutes a "legal" instance of the model; any node can be connected to any other node irrespective of its label, which is assigned randomly. Fig. 4 shows an example of a random graph generated by our procedure.

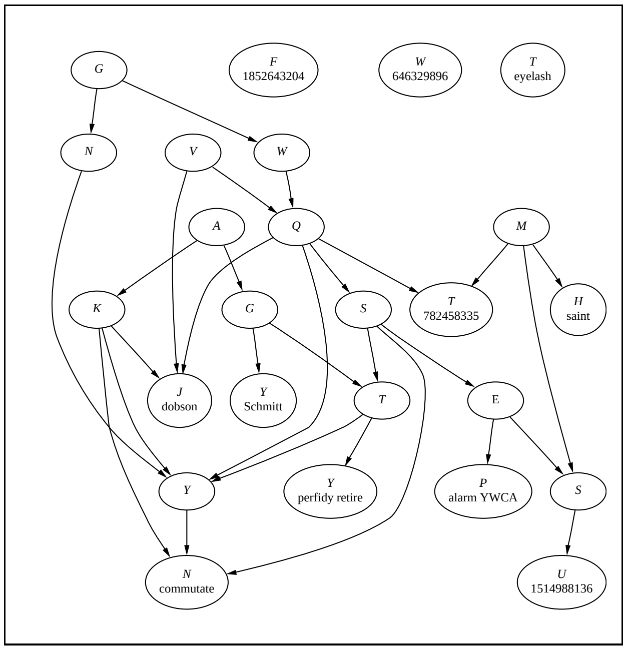

Figure 4: An instance of a random graph.

4 Experimental Results

To test the concept of graph probing, we designed a series of simulation studies as well as an experiment using real results from a commercial OCR system. As indicated, we would like to be able to equate probing agreement (i.e., Eq. (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. In the case of the OCR experiment where there does already exist a practical, accepted methodology for computing accuracy, string edit distance, we correlate probing agreement with normalized edit distance.

In the studies that follow, it is important to keep in mind that the probes are always generated automatically, working directly from the recognition result and the ground-truth. Once the probe classes have been defined (which need only be done once), graph probing is a completely autonomous evaluation paradigm.

4.1 Simulation Results for the Entity Graph Model

Our procedure begins by creating a graph for a page with a random number of zones (all random quantities in our simulations are chosen uniformly from within a specified range). 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, Class 1 and Class 2 probes, and gathering relevant statistics. The study required about 4.5 hours to run on an SGI O2 workstation.

The results for the entity graph experiment are presented in Tables 1 and 2 and in Fig. 5. As can be seen from the first table, there was a wide range in the size of the graphs under consideration. In terms of probes, those from Class 1 were by far the most prevalent (the other two classes sum the results for all nodes in certain broad categories: having the same label or the same in-/out-degree). On average, approximately one probe was generated for each node, and each pair of graphs required about half a minute to compare via probing.[2] Overall, the average probing agreement was 0.853, and the maximum was 0.996 (i.e., the probes always captured the fact that one of the graphs contained errors).

The ability of the three probe classes to differentiate the two graphs is shown in Table 2. Class 1 probes never failed in this experiment. Note that, by definition, Class 0 and Class 2 probes will always miss differences that involve only content, but various offsetting combinations of edits have the potential to confuse any of the classes. The last column in Table 2 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 the number of graph editing operations is shown in the chart in Fig. 5. The datapoints show the average at each step along the x-axis, while the vertical bars give the min/max range. Turning this around, it can be seen that the size of the discriminating probe set provides a reasonably dependable measure of the difference between two graphs. It seems likely that refining and/or weighting the probe sets appropriately could lead to an improvement in the "outliers;" this is a topic for future research.

[2] As noted earlier, our probes are written in an extension of Tcl/Tk, an interpreted scripting language. In a "production" environment, a more efficient implementation could be achieved using a compiled language.

Table 1: Statistics for the entity graph experiment (500 random graphs).

Attribute	*Min*	*Max*	*Ave*
Zones	1	8	4.9
Lines	1	54	21.7
Words	2	278	108.4
Nodes	5	341	136.0
Edits	1	57	13.3
Class 0 Probes	8	8	8.0
Class 0 Agreement	0.250	1.000	0.530
Class 1 Probes	9	522	216.9
Class 1 Agreement	0.000	0.998	0.912
Class 2 Probes	10	39	29.4
Class 2 Agreement	0.000	1.000	0.576
Overall Probes	28	561	254.3
Overall Agreement	0.177	0.996	0.853
Probes/Node	0.866	1.750	0.985
Probe Time (secs)	0.780	145.010	32.791
Secs/Probe	0.025	0.258	0.107

Table 2: Performance by probe class for the entity graph experiment (500 random graphs).

Probes	*Detected*	*Missed*	*% Detected*	*Unique*
Class 0	450	50	90.0%	0
Class 1	500	0	100.0%	34
Class 2	466	34	93.2%	0
Overall	500	0	100.0%	n/a

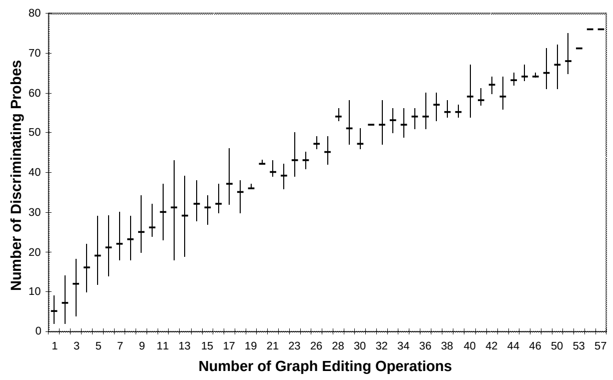

Figure 5: Discriminating probes as a function of edits for the entity graph experiment (500 random graphs).

4.2 Simulation Results for the Table Graph Model

Like 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 chosen from the **spell** dictionary,

Table 3: Statistics for the table graph experiment (500 random graphs).

Attribute	*Min*	*Max*	*Ave*
Rows	2	15	8.5
Cols	2	6	4.0
Nodes	8	111	46.7
Edits	1	24	9.8
Class 0 Probes	6	6	6.0
Class 0 Agreement	0.000	1.000	0.362
Class 1 Probes	8	174	68.3
Class 1 Agreement	0.154	0.986	0.811
Class 2 Probes	4	6	5.9
Class 2 Agreement	0.000	1.000	0.122
Class 3 Probes	8	174	67.5
Class 3 Agreement	0.000	0.943	0.641
Overall Probes	26	360	147.7
Overall Agreement	0.056	0.966	0.686
Probes/Node	1.231	1.674	1.561
Probe Time (secs)	0.870	178.610	32.793
Secs/Probe	0.032	0.496	0.164

Table 4: Performance by probe class for the table graph experiment (500 random graphs).

Probes	*Detected*	*Missed*	*% Detected*	*Unique*
Class 0	447	53	0.894	0
Class 1	500	0	1.000	0
Class 2	440	60	0.880	0
Class 3	500	0	1.000	0
Overall	500	0	1.000	n/a

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, deleting a *Row* or *Column* node (along with all of its associated *Cell* nodes), or inserting a new *Row* or *Column*. In addition to the Class 0, 1, and 2 probes of the first study, we also include the Class 3 probes described in subsection 3.2.

Tables 3 and 4 and Fig. 6 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 nearly identical owing to the new probe class. As Table 3 indicates, the Class 1 and 3 probes never failed. Overall, the average agreement was found to be 0.686.

The detection capabilities of the various probe classes are listed in Table 4. That the Class 0 and 1 probes exhibit roughly the same number of misses as they did in Table 2 is coincidental (this depends on the distributions of the random edits used). Note that there was no instance where a single 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 Fig. 6. 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.

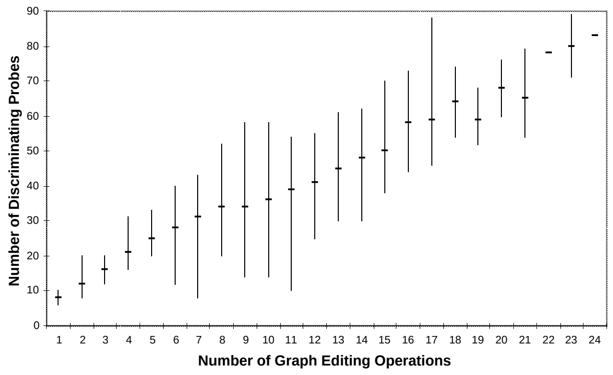

Figure 6: Discriminating probes as a function of edits for the table graph experiment (500 random graphs).

4.3 Simulation Results for the Random Graph Model

The previous two studies were restricted in terms of the initial graphs and the permissible edits that could be performed on them; the graphs always had to be legal instances of an entity or table graph. In this final simulation, the generated graphs are completely unconstrained random DAG's. Node labels are chosen from among the 26 upper case letters A-Z. Certain nodes are designated as leaf nodes; these are assigned random content (words or integers). The remainder of the nodes are attached to varying numbers of children. As was depicted in Fig. 4, the graphs need not be fully connected. The set of possible editing operations consists of changing node labels or content, deleting or inserting nodes, and deleting or inserting edges.

Results for 500 random graphs are given in Tables 5 and 6 and Fig. 7. There were, on average, 251 nodes and 328 edges in the graphs in this study. As Table 5 shows, the average overall agreement was 0.872, while the maximum agreement was 0.966 (i.e., the fact that two graphs were different was detected without fail when all probe classes were taken into account).

The ability of each class to detect the differences is shown in Table 6. The best-performing class (Class 2) contained at least one discriminating probe 97% of the time, while the worst (Class 1) was successful 89% of the time. Perhaps the most important conclusion to be drawn from this table is that none of the classes was redundant; each of them detected at least one case that the other two classes missed.

The plot of discriminating probes versus graph

Table 5: Statistics for the random graph experiment (500 random graphs).

Attribute	*Min*	*Max*	*Ave*
Nodes	73	251	164.4
Edges	58	328	180.9
Edits	1	25	11.5
Class 0 Probes	46	52	51.6
Class 0 Agreement	0.520	1.000	0.841
Class 1 Probes	50	322	165.2
Class 1 Agreement	0.783	1.000	0.962
Class 2 Probes	28	56	42.3
Class 2 Agreement	0.171	1.000	0.570
Overall Probes	140	413	259.1
Overall Agreement	0.620	0.993	0.872
Probes/Node	0.577	1.142	0.816
Probe Time (secs)	3.070	249.940	17.926
Secs/Probe	0.019	1.032	0.069

Table 6: Performance by probe class for the random graph experiment (500 random graphs).

Probes	*Detected*	*Missed*	*% Detected*	*Unique*
Class 0	474	26	0.948	1
Class 1	447	53	0.894	8
Class 2	486	14	0.972	6
Overall	500	0	1.000	n/a

editing operations shown in Fig. 7 bears a strong resemblance to those for the previous two simulations (Figs. 5 and 6). This provides support for our belief that graph probing is a general evaluation paradigm that can be applied across a range of applications that employ graph representations.

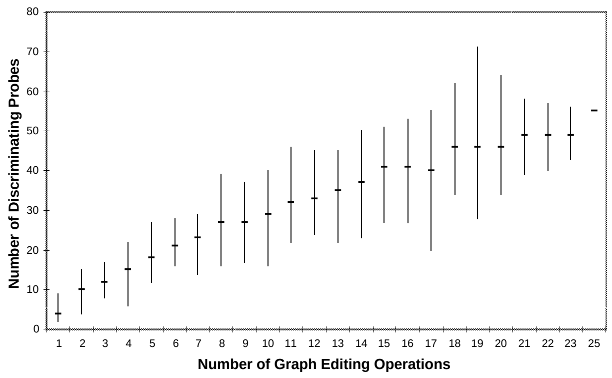

Figure 7: Discriminating probes as a function of edits for the random graph experiment (500 random graphs).

4.4 Experimental Results for Output from an OCR System

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 [10, 11]). 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. Instead, we designed an experiment making use of the output from a commercial OCR system, post-processed in the obvious way to yield a graph employing the lower two levels of the entity graph model (i.e., *Line* and *Word* nodes). Since string edit distance is an accepted methodology for evaluating OCR results [7], we have access to a standard to which to compare graph probing.

Table 7: Accuracies for the OCR experiment (60 document pages).

	OCR Accuracy		
Document Type	*Min*	*Max*	*Ave*
Printed	93.9%	96.7%	95.8%
Faxed	60.7%	88.4%	73.7%
3rd Generation	63.4%	93.2%	80.7%
Light	78.8%	89.8%	84.6%
Dark	92.6%	96.6%	95.5%
Annotated	56.3%	83.3%	74.9%

The test collection consisted of 10 professionally written news articles gathered from Usenet, ranging in length from 91 to 379 words. For each document, six different versions were created, each formatted in 11-point Times font with a 13-point line spacing under Microsoft Word. One copy of each page was printed and then scanned at 300 dpi using a UMAX Astra 1200S scanner. The remaining five versions of the page were subjected to one of five different degradations before scanning: faxing, noticeably light or dark or third generation photocopying, or handwritten annotation ("redacting") that obscured a randomly chosen 20% of the lines on the page. All of the page images were then OCR'ed using Caere OmniPage Limited Edition.

Normalized string edit distance was used to compute the OCR accuracies [7]. The minimum, maximum, and average accuracies for the 10 pages of a given type are listed in Table 7. As can be seen, some of the documents experienced severe damage, yielding a wide range of accuracies (dropping from 96.7% down to 56.3%). In addition to the many expected character misrecognitions (which induce word-level errors in the entity graph representation), this particular OCR system attempts to concatenate text lines that it believes fall logically within the same paragraph. This policy leads to the potential for disagreements at the *Line* level in the entity graph model as well.

Basic statistics for the probing evaluation of the 60 test pages are presented in Table 8. There were errors in every one of the recognized pages, and this is reflected in the maximum overall agreement which is 0.963. As in the simulations, approximately one probe was generated for each node in the graphs (under the current definitions of the probe sets, this

Table 8: Statistics for the OCR experiment (60 document pages).

Attribute	*Min*	*Max*	*Ave*
Zones	1	1	1.0
Lines	2	43	16.1
Words	42	379	195.4
Nodes	48	424	213.5
Class 0 Probes	8	8	8.0
Class 0 Agreement	0.500	1.000	0.604
Class 1 Probes	107	758	390.8
Class 1 Agreement	0.371	0.998	0.754
Class 2 Probes	17	40	27.5
Class 2 Agreement	0.051	1.000	0.173
Overall Probes	134	804	426.3
Overall Agreement	0.359	0.963	0.709
Probes/Node	0.965	1.107	1.011
Probe Time (secs)	7.210	269.600	90.585
Secs/Probe	0.054	0.337	0.177

Table 9: Performance by probe class for the OCR experiment (60 document pages).

Probes	*Detected*	*Missed*	*% Detected*	*Unique*
Class 0	59	1	0.983	0
Class 1	60	0	1.000	1
Class 2	59	1	0.983	0
Overall	60	0	1.000	n/a

quantity is tied strongly to the number of words in the two documents in question). The probing time averaged 90 seconds per document page. While this is significantly longer than the time needed to compute simple string edit distance, one must remember that graph probing is a more general measure, capable of detecting errors in logical structure as well as in content.

Table 9 shows that all of the probe classes were capable of detecting that there were differences between the ground-truth and recognized documents in nearly every case (recall that the OCR accuracies ranged as high as 96.7%). Only in one instance did the Class 1 probes outperform the other two.

The remaining issue, then, is seeing how well graph probing correlates with traditional string edit distance. These results are plotted in Fig. 8. Here we show a distinct style of datapoint for each of the six kinds of copies in the test set. This sort of evaluation is not a particularly fair test for graph probing as the entity graph model we are currently using is word- and not character-based (the model can only distinguish between "zero" and "one or more" errors in a word – it cannot count errors). Even so, while the correspondence between the two measures is somewhat hazier than in the simulations, an overall-monotonic behavior is still visible.

5 Conclusions

This paper has described an intuitive, easy-to-implement scheme for the problem of performance

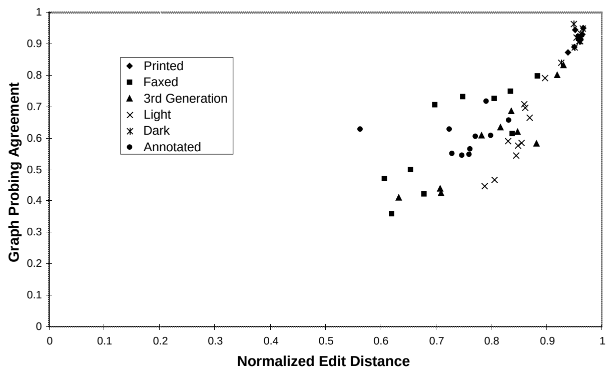

Figure 8: Graph probing agreement as a function of edit distance for the OCR experiment (60 document pages).

evaluation when document recognition results are represented in the form of a directed acyclic graph. Graph probing can be seen as having its roots in past work on heuristics for solving the graph isomorphism problem, however its utility extends beyond simply testing graphs for equivalence; it also allows us to quantify the similarity between two graphs. We presented results from three simulation studies using different graph models and an experiment employing real OCR data to demonstrate the applicability of the approach.

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 defined it in Section 2 is an off-line procedure (i.e., all of the probes are computed in advance, before the first probe is evaluated). Allowing the probing to take place on-line, making it adaptive, might add significant power.

Lastly, other applications could make use of this technique for graph comparison. In information retrieval, for example, queries and target documents can sometimes be represented in terms of graphs (e.g., HTML parse trees).

6 Acknowledgements

Jianying Hu and Ramanujan Kashi played important roles in the table recognition research which lead to the development of the graph probing paradigm. The graph drawings shown in this paper were generated using the **dot** tool built by Eleftherios Koutsofios and Stephen C. North [15]. The trademarks mentioned in this paper are the properties of their respective companies.

References

[1] M. Abdulrahim and M. Misra. A graph isomorphism algorithm for object recognition. *Pattern Analysis and Applications*, 1(3):189–201, 1998.

[2] S. Agne, M. Rogger, and J. Rohrschneider. Benchmarking of document page segmentation. In *Proceedings of Document Recognition and Retrieval VII (IS&T/SPIE Electronic Imaging)*, volume 3967, pages 165–171, San Jose, CA, January 2000.

[3] L. Babai, P. Erdös, and S. M. Selkow. Random graph isomorphism. *SIAM Journal on Computing*, 9(3):628–635, August 1980.

[4] D. G. Corneil and D. G. Kirkpatrick. A theoretical analysis of various heuristics for the graph isomorphism problem. *SIAM Journal on Computing*, 9(2):281–297, May 1980.

[5] G. Dudek, P. Freedman, and S. Hadjres. Using local information in a non-local way for mapping graph-like worlds. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1639–1645. Morgan Kaufmann, August 1993.

[6] G. Ellis and F. Lehmann. Exploiting the induced order on type-labeled graphs for fast knowledge retrieval. In *Proceedings of the 2nd International Conference on Conceptual Structures, Lecture Notes in Artificial Intelligence, Number 835*, pages 293–310. Springer-Verlag, August 1994.

[7] J. Esakov, D. P. Lopresti, J. S. Sandberg, and J. Zhou. Issues in automatic OCR error classification. In *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval*, pages 401–412, Las Vegas, NV, April 1994.

[8] S. Fortin. The graph isomorphism problem. Department of Computer Science Technical Report TR 96-20, The University of Alberta, July 1996.

[9] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, San Francisco, CA, 1979.

[10] 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.

[11] J. Hu, R. Kashi, D. Lopresti, and G. Wilfong. Table structure recognition and its evaluation. In *Proceedings of Document Recognition and Retrieval VIII (IS&T/SPIE Electronic Imaging)*, volume 4307, pages 44–55, San Jose, CA, January 2001.

[12] Y. Ishitani. Model matching based on association graph for form image understanding. In *Proceedings of the Third International Conference on Document Analysis and Recognition*, pages 287–292, Montréal, Canada, August 1995.

[13] J. Kanai. Automated performance evaluation of document image analysis systems: Issues and practice. *International Journal of Imaging Science and Technology*, 7:363–369, 1996.

[14] T. Kanungo, C. H. Lee, J. Czorapinski, and I. Bella. TRUEVIZ: a groundtruth / metadata editing and visualizing toolkit for OCR. In *Proceedings of Document Recognition and Retrieval VIII (IS&T/SPIE Electronic Imaging)*, volume 4307, pages 1–12, San Jose, CA, January 2001.

[15] E. Koutsofios and S. C. North. Drawing graphs with dot. Technical Report 59113-910904-08TM, AT&T Bell Laboratories, September 1991.

[16] B. McKay. *Nauty User's Guide (Version 1.5).* Computer Science Department, Australian National University.

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

[18] B. T. Messmer and H. Bunke. Efficient error-tolerant subgraph isomorphism detection. In D. Dori and A. Bruckstein, editors, *Shape, Structure and Pattern Recognition*, pages 231–240. World Scientific, Singapore, 1995.

[19] G. Nagy. Document image analysis: Automated performance evaluation. In A. L. Spitz and A. Dengel, editors, *Document Analysis Systems*, pages 137–156. World Scientific, Singapore, 1995.

[20] J. K. Ousterhout. *Tcl and the Tk Toolkit.* Addison-Wesley, Reading, MA, 1994.

[21] C. Peterman, C. H. Chang, and H. Alam. A system for table understanding. In *Proceedings of the Symposium on Document Image Understanding Technology*, pages 55–62, Annapolis, MD, 1997.

[22] A. Sanfeliu and K.-S. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(3):353–362, May/June 1983.

[23] R. J. Schalkoff. *Pattern Recognition: Statistical, Structural and Neural Approaches*, chapter 8. John Wiley & Sons, New York, NY, 1992.

[24] L. Shapiro and R. M. Haralick. A metric for comparing relational descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7(1):90–94, January 1985.

[25] G. Tinhofer and M. Klin. Algebraic combinatorics in mathematical chemistry. methods and algorithms. iii. graph invariants and stabilization methods. Technical Report TUM M9902, Techn. Univ. München, March 1999.

[26] B. A. Yanikoglu and L. Vincent. Groundtruthing and benchmarking document page segmentation. In *Proceedings of the Third International Conference on Document Analysis and Recognition*, pages 601–604, Montréal, Canada, August 1995.