2009 10th International Conference on Document Analysis and Recognition

An RDF-Based Blackboard Architecture for Improving Table Analysis

Vanessa Long
Macquarie University
Sydney, Australia
vanessa@ics.mq.edu.au

Abstract

Table analysis is a complex problem, involving searching solutions from a large search space. Studies show that finding the most credible answers to complex problems often require combining multiple kinds of knowledge. Although the literature shows that both layout and language information have been used in table extraction systems, the amount of information each system uses is limited, and up till now, there is not an easy, systematic way to incorporate new information in these systems. This paper describes a framework for combining multiple solutions (including partial solutions) to solve a general table recognition problem.

Keywords: Blackboard architecture, multi-agent system, intelligent experts, table analysis

1. Introduction

Due to their expressive power, tables are often the preferred means of storing, representing and communicating structured data. The richness and the important nature of information contained within tables makes table processing an important research area. Over the years, many table analysis systems have been produced. Most systems rely on certain features to find the solutions to the problems they try to solve. For example, a system might use the number of blank lines to identify table boundaries, as blank lines are often used to separate table and non-table regions. However, this feature produces correct and incorrect answers at the same time, as blank lines are not used to separate table and non-table regions exclusively. Because using a small set of features often lead to unreliable answers, it is often necessary to incorporate multiple sources of evidence in a problem-solving process.

Using table boundary identification from plain text documents as an example, this paper shows how the *blackboard architecture*, which supports the use of multiple knowledge sources through autonomous programs called *agents* (also known as *experts*) in a problem-solving process, can help to solve a general table analysis problems.

2. An RDF-Based Blackboard Framework for Table Analysis

The blackboard architecture, which was first published in 1980 when the HEARSAY-II speech understanding system [2, 3] was built, can be understood as a global memory space (the blackboard) being provided to a group of agents who try to solve a problem collaboratively. The idea is that although no agent has the entire solution to a complex problem, because each agent possesses only limited knowledge about the overall solution, the overall solution might be found in the combined knowledge of all the agents.

The speech understanding problem and table analysis problems are similar in that there is not a single algorithm that can deterministically produce all the solutions, and finding solutions to these problems involves searching through a large space of candidate solutions. Since the speech understanding problem was solved with a high level of accuracy under the blackboard architecture, it was hypothesised that the blackboard architecture can help solving table analysis problems with satisfactory results.

The blackboard architecture is characterised by a *discussion* phase, during which multiple (possibly heterogeneous) agents contribute their knowledge to the solution of the problem, followed by the *conflict resolution phase*, where a single agent resolves conflicting opinions produced by the agents (if there are any), and produces the overall solution. During the discussion phase, blackboard access is given to agents sequentially. Each agent in the blackboard architecture bears both opportunistic and knowledge-sharing characteristics. Being opportunistic, as soon as the blackboard contains enough information for an agent to produce results (even just partial results), it would do so without delay. Being knowledge-sharing, each agent publishes its knowledge on the blackboard.

It is worth noting that an agent might not know what to write until it sees what other agents have written on the blackboard. Because of this, agents' contributions to the overall solution can be incremental - it can take a few rounds of blackboard passing before the overall solution can be constructed.

978-0-7695-3725-2/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDAR.2009.81

The blackboard architecture is more suitable for combining multiple sources of knowledge than non-blackboard architecture (i.e. the subroutine-like architecture). The main reason for this is that there is no need to schedule the running sequence of agents under the blackboard architecture. Agents' activation is determined by the information available on the blackboard, rather than by explicit calls from other agents or some central sequencing mechanism. As a result, programs do not need to decide whether or when a module calls another module at design time. Agents in the subroutine-like architecture, on the other hand, directly call each other. This requires built-in knowledge about the agents and their relationships in a system when introducing new agents [2, 3].

This characteristic makes it easy to add or to remove knowledge sources in the blackboard architecture. This, in turn, results in several other benefits, which include encouraging the development of diverse knowledge sources, the inclusion of partial knowledge in a problem-solving process, and the ease of experiments and evaluation of individual agents.

Before a blackboard system can be implemented, a number of key design decisions must be made: what can be written on the blackboard, whether agents allowed to remove statements from a blackboard, how to coordinate agents' actions, how to detect the end of a discussion phase, and how to detect and resolve conflicts among agents. The following section provides answers to these questions in relation to the blackboard framework used in this paper.

In this paper, the content of the blackboard is restricted to *RDF* (Resource Description Framework) statements using a predefined vocabulary and concepts, such as 'number of columns', 'content of a table cell', and 'table cell row index'. RDF has been adopted as the standard of knowledge representation by the semantic web community, and details can be found in many textbooks. The main reasons for choosing RDF are that RDF has the ability to allow any relationships between any parts of tables to be established, and RDF has the ability to refer to low-level content, such as the positions of characters in documents, as well as high-level content, such as descriptions of tables at the same time. For example, the statement, *'the character at line 9 and column 50 is c'*, and the statement, *'the table contains a calculation'* can both be represented using RDF syntax. These abilities make RDF suitable for integrating table structural analysis with semantic analysis, as statements about geometric relations between elements and statements about functional relations can be expressed at the same time.

There are pros and cons associated with the decision of allowing (or not allowing) statements to be deleted from a blackboard. On the one hand, if deletion is not allowed, then the blackboard content might grow to a size that is too big to be efficient. Sometimes a blackboard may contain erroneous information. By allowing deletion, errors can be removed from the blackboard. On the other hand, allowing deletion can cause loss of information, and this may result in agents entering into one type of infinite loop: a message that is written by an agent is deleted by another agent, and the absence of the message makes the first agent write the message again. To balance the pros and cons, the blackboard content deletion policy used in this paper is to allow the conflict resolver (an agent) to delete conflicting statements, but the same privilege is not given to other agents.

Due to their opportunistic character, two or more agents could fight for blackboard access if they all have information to contribute. Therefore, a control unit is needed to coordinate the agent operations. In this paper, the running order of the agents is determined by a control unit called the *scheduler*, whose main functions are to mediate among agents competing to access the blackboard, and to drive the problem solving process towards finding the overall solution.

For simplicity, the scheduler in this paper passes the blackboard to agents in a round robin fashion until their discussion ends. Before the scheduler passes the blackboard to an agent, it takes a snapshot of the blackboard. When an agent finishes accessing the blackboard, the blackboard is returned to the scheduler. Upon receiving the blackboard, the scheduler compares the blackboard content to decide whether the agent has deleted any statements, or whether the agent has written any new statements. If an agent deletes information from the blackboard when it is not supposed to, the scheduler can restore the blackboard content before passing it to the next agent. If no new statement is produced after passing the blackboard to every agent, then the scheduler concludes that the discussion among agents has ended, and the blackboard is given to agents for resolving conflicts (if there are any) and finding the overall solution.

While agents work towards a goal, it is possible that they will disagree with each other. Resolving conflicts is one of the characteristics of a blackboard system. In the experiments covered in this paper, conflicting opinions can occur in two cases: a text line is said to be a table-line and a not-table-line at the same time, and a text line is said to mark a table-begin region and a table-end region at the same time. In the first case, the conflict resolver will resolve the text line to be a neutral-line (that is, it is neither a table-line nor a not-table-line). In the second case, the conflict resolver will resolve the text line as neither a table-begin boundary nor a table-end boundary. Once conflicts are resolved, an agent then can try to produce the overall solution based on the blackboard content. The next section shows how the blackboard architecture can be applied to identify table boundaries.

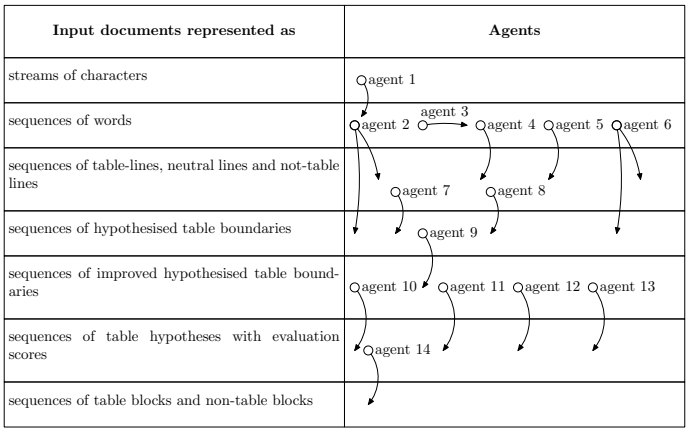

Figure 1. Participating agents for the experiments in section 3.

3. Applying the Blackboard Framework to Table Boundary Identification

In this paper, the table boundary identification problem is decomposed into two sub-problems. The first one is to decide whether a line in an input document is a table-line or not. There are three possible outcomes: yes, no, and not sure. The second one is to identify table boundaries based on the results from the first sub-problem.

Initially, input documents are represented as streams of ASCII characters. As the problem-solving processes advance, input documents are represented at a number of progressing levels: the word level, the line level, and ultimately, the table level. Figure 1 shows this decomposition in terms of levels of information needed to be processed. It also shows the agents involved at each level of information processing. The levels where the arrows originating from indicate the input levels, and the levels where the arrows terminating indicate the output levels.

The blackboard framework discussed in this paper has the ability to integrate existing and future knowledge sources. To show this, 14 knowledge sources (including 5 from the literature), 1 property inferencing agent and 1 conflict resolving agent are used in the experiments. The 5 agents from the literature are agents #2, #6, #10, #11, and #12 in Figure 1. The reasons for including the above knowledge sources are: they are frequently used in detecting tables by various researchers; the literature has provided sufficient details for the information to be implemented as agents from their written descriptions; they represent various kinds of information; the algorithms they use are of various levels of complexities; and they cover various programming paradigms such as deterministic methods, machine learning, and grammar rule based algorithms. The functional description of the agents are provided below.

Agent #1's goal is to post input documents, word by word, onto the blackboard. For each word, the agent writes the word's line and column indices on the blackboard. The agent also tries to recognize certain numeric word types, such as 'positive decimal number' and 'percentage', and writes the information on the blackboard.

Agent #2 tries to identify tables using the context-free grammar based on the one published by Long et al. [7]. Information posted on the blackboard by this agent includes the document width, length, information about the location and the length of each input line, and locations of the table blocks.

Agent #3 identifies neighbouring text line. For each text line (that is, a line that is neither a blank line nor a ruling line) in a document, agent #3 identifies its following text lines and posts their distances (in terms of number of lines) on the blackboard.

Agent #4 identifies not-table-line based on line length. According to this agent, a text line is a not-table line if one of the following conditions is met: (1) it does not contain three or more consecutive space characters, and its length is at least 90% of the document width; (2) its lengths is at least 50% of the document width, and both the text line and its its immediately following text line do not contain three or more consecutive space characters.

Agent #5 identifies not-table-line based on surrounding line types. This agent possesses the knowledge that a table-line should not appear in the middle of a not-table block. In particular, if a text line is tagged as a table-line, but its two immediately preceding text lines and its two immediately following text lines are identified as not-table lines, then the agent will disagree with the previous judgement and tag the line as a not-table line.

Agent #6 identifies tables based on the benchmark algorithm published by Ng et al. [8]. Since Ng et al.'s paper only specifies how to identify table-lines, and it does not specify how to identify table boundaries, it is assumed that a table is formed by 2 or more consecutive table-lines, where pair of adjacent table-lines can be separated by 1 blank line. Information posted on the blackboard by this agent includes the classification of each input line, the table-begin and table-end boundaries of each table block.

Agent #7 detects table boundaries based on the change of layout patterns in neighbouring lines and the appearance of certain keywords. In particular, it tests the following conditions to find table-begin and table-end boundaries: a table-line is a table-begin boundary if it is the first text line in a document; similarly, a table-line is a table-end boundary if it is the last text line in a document; a table-line is a table-begin boundary if it contains one or more frequently used header keywords such as *'AUD000'*, *'$000'*, *'$thousand'*, *'$million'* and dates; a table-line is a table-begin boundary if its preceding two text lines are not-table-lines, and

if its following two text lines are table-lines; and similarly, a table-line is a table-end boundary if its preceding two text lines are table-lines, and if its following two text lines are not-table-lines.

Agent #8 identifies table-begin boundaries from vertically aligned numeric sequences in neighbouring text lines. According to this agent, the first table-line that contains vertically aligned numeric sequences is identified as table-begin boundary. Additionally, a table-line is identified as a table-begin boundary if it does not contain any numeric sequences, and is directly above a block of table-lines that contain vertically aligned numeric sequences.

Agent #9 possesses the knowledge that table-begin and table-end boundaries should appear in an alternating sequence. The agent tries to insert a table-end boundary between two consecutive table-begin boundaries using this method: starting from the second table-begin boundary, or the first not-table line between the two table-begin boundaries (whichever appears first), all the way back to the third line after the first table-begin boundary, the first table-line is identified as a table-end boundary. Using similar method, the agent tries to identify a table-begin boundary between two consecutive table-end boundaries.

Given a table hypothesis, agent #10 evaluates its column structure using the hierarchical clustering algorithm and a set of heuristic rules published by Hu et at. [5, 6]. Information posted on the blackboard by this agent includes the number of columns in a table hypothesis, and the data type consistency in each column.

Given a table hypothesis, agent #11 implements the table scoring algorithm published by Hu et al. [4]. The scores are based on character correlations between vertically aligned characters. Information posted on the blackboard by this agent includes a score measuring the correlation of character types among vertically aligned characters in a table hypothesis.

Given a table hypothesis, agent #12 reports the header information in a hypothesis based on what is found in the literature [1, 9, 10, 11]. Information posted on the blackboard by this agent include the number of lines containing typical headers, the distance between the header lines, the maximum number of consecutive spaces in a line, and the number of lines not containing two or more space characters.

Agent #13 posts the number of content line in a table hypothesis on the blackboard. All other things being equal, tables with larger number of content lines are preferred to avoid partial table detection.

Agent #14 assembles the overall solution based on the evaluators' reports. The agent first judge whether a table hypothesis should be accepted or not. It does this by *learning* what a table should look like by applying the J48 decision tree classifier algorithm to the information posted by agents #10, #11, and #12. The learnt knowledge, which is a decision tree, is then used to accept or reject a table hypothesis. If the hypothesis is accepted, then it is assigned with the score produced by agent #13; otherwise, it is assigned with an arbitrary low score (say -1000). After assigning table hypotheses with scores, the agent then goes through an optimisation process for selecting the final answer so that the total score of selected tables is maximised. Final decision of the table-begin and table-end boundaries in a document is posted on the blackboard by this agent.

4. Evaluation

Experiments in this paper compare a multi-agent table boundary identification system using the agents discussed in section 3 against the benchmark performance produced by agent #6.

4.1. Evaluation Measures

Recall and precision have been the mainstream reporting measures. However, quoting recall and precision alone does not provide feedback on the types of errors and the sizes of errors. For example, missing one line of an expected answer and almost all lines of the expected answer would result in the same number of errors. To overcome this weakness, the evaluation measures covered in this paper use bounding boxes to represent table components at table-level, row-level and cell-level. The intersecting patterns between the rectangles representing the experimental results and the rectangles representing the expected answers are reported (see Table 1). The overall score is calculated as the overlapping area between expected answers and experimental results divided by the total area between expected answers and experimental results. The evaluation measures in Table 1 show that the experimental result discussed in section 3 has a much higher overall score at the table-level than the ones at the row-level and cell-level. This is due to the fact that the experiment was designed to identify table boundary rather than to work out table structures.

Additionally, input documents that contain no tables can only be used to test insertion errors. To disclose the quality of test data, both the number of test cases used for evaluation, and the portion of the test cases that does not contain tables should be reported. In this paper, 275 (78.12%) of the 352 documents in the test set do not contain any table.

4.2. Test Data

In this paper, the experiments are conducted on data sampled from the ASX (Australian Stock Exchange) corpus, which contains financial reports (in plain text format) submitted to the ASX between 2001 and 2004 by public companies in Australia. The test data consists of 346 randomly sampled documents from the corpus and 6 manually selected documents that contain tables that are considered to be relatively difficult to detect.

Experiment	Insertion errors on test cases without tables	Test cases that contain tables							Area of bounding boxes being detected correctly
		Number of expected answers	Number of correctly detected tables	Non-overlapping error		Overlapping error			
				Insertion	Deletion	Proper subset error	Proper superset error	Interset error	
Benchmark	0	170	11	166	40	81	0	38	7.63%
Blackboard	2	170	30	21	13	15	100	12	70.82%

Proper subset error: parts of expected answer are detected without introducing not-table contents
Proper superset error: the whole expected answer tables, together with some not-table contents, are detected
Interset error: parts of expected answer, together with some not-table contents, are detected
The benchmark result is produced by the deterministic method published by Ng et al. [8]

Table 1. Comparison between the performance produced by the benchmark algorithm published in [8] and the performance produced by the agents described in section 3.

4.3. Evaluation Results

The evaluation results in Table 1 confirm the observation set out in section 1, which suggests that the use of multiple kinds of knowledge can help to improve table recognition performance. In the experiment, the blackboard architecture has played an important role in combing multiple kinds of knowledge.

5. Conclusion and Future Work

This paper discusses a multi-agent framework for combining multiple knowledge sources in a general table processing system. Through experiments, the benefits coming from the ease of combining various knowledge sources given by the blackboard architecture are seen. Although experiments were conducted on identifying table boundaries from plain text documents, it is not difficult to see that the blackboard framework can be applied to a general table processing task.

The blackboard architecture provides the flexibility of incorporating various knowledge sources in table processing tasks. However, the decision of using the blackboard architecture alone does not guarantee high levels of accuracy. There are many other factors, such as the way a problem is decomposed and the effectiveness and robustness of the knowledge sources, that determine the accuracy of a table processing task. Since whether a problem can be solved effectively depends ultimately on the capability of individual agents, finding the right techniques and improving effectiveness of knowledge sources will remain the focus of future research directions.

References

[1] D. W. Embley, C. Tao, and S. W. Liddle. Automatically extracting ontologically specified data from html tables with unknown structure. In *Proceedings of the 21st International Conference on Conceptual Modeling (ER 2002)*, pages 322–327, Tampere, Finland, October 2002.

[2] L. D. Erman, F. Hayes-Roth, V. R. Lesser, and D. R. Reddy. The hearsay-ii speech understanding system: Integrating knowledge to resolve uncertainty. *Computing Surveys*, 12(2):213–253, 1980.

[3] L. D. Erman and V. R. Lesser. *The HEARSAY-II Speech Understanding System: A Tutorial*, pages 235–245. Morgan Kaufmann, 1990.

[4] J. Hu, R. Kashi, D. Lopresti, and G. Wilfong. Medium-independent table detection. In *Document Recognition and Retrieval VII*, pages 291–302, 2000.

[5] J. Hu, R. Kashi, D. Lopresti, and G. Wilfong. Experiments in table recognition. In *Workshop on Document Layout Interpretation and Applications*, Seattle, Washington, 2001.

[6] J. Hu, R. Kashi, D. Lopresti, and G. Wilfong. Table structure recognition and Its Evaluation. In *Document Recognition and Retrieval VIII*, pages 44–55, 2001.

[7] V. Long, R. Dale, and S. Cassidy. A model for detecting and merging vertically spanned table cells in plain text documents. In *Proceedings of the 8th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1242–1246, Seoul, Korea, September 2005. IEEE.

[8] H. T. Ng, C. Y. Lim, and J. L. T. Koo. Learning to recognize tables in free text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 443–450, College Park, Maryland, USA, 1999.

[9] D. Pinto, A. McCallum, X. Lee, and W. B. Croft. Table extraction using conditional random fields. In *Proceedings of the 26th ACM SIGIR*, pages 235–242, Toronto, Canada, 2003.

[10] K. M. Tubbs and D. W. Embley. Recognizing records from the extracted cells of microfilm tables. In *ACM Symposium on Document Engineering*, pages 149–156, McLean, Virginia, 2002.

[11] X. Wei, B. Croft, and A. McCallum. Table extraction for answer retrieval. In *Information Retrieval*, pages 589–611. Springer Netherlands, September 2006.