Table Extraction for Answer Retrieval

Xing Wei, Bruce Croft, Andrew McCallum

Center for Intelligent Information Retrieval

University of Massachusetts Amherst

140 Governors Drive

Amherst, MA 01003

{xwei, croft, mccallum}@cs.umass.edu

Abstract

The ability to find tables and extract information from them is a necessary component of many information retrieval tasks. Documents often contain tables in order to communicate densely packed, multi-dimensional information. Tables do this by employing layout patterns to efficiently indicate fields and records in two-dimensional form. Their rich combination of formatting and content presents difficulties for traditional retrieval techniques. This paper describes techniques for extracting tables from text and retrieving answers from the extracted information. We compare machine learning (especially, Conditional Random Fields) and heuristic methods for table extraction. To retrieve answers, our approach creates a cell document, which contains the cell and its metadata (headers, titles) for each table cell, and the retrieval model ranks the cells of the extracted tables using a language-modeling approach. Performance is tested using government statistical Web sites and news articles, and errors are analyzed in order to improve the system.

Keywords

Table extraction, conditional random fields, question answering, information extraction

1. INTRODUCTION

Information in many documents is carried not only in their stream of words, but also by the layout of those words. Just as prepositions serve to make relations between phrases, two-dimensional layout also serves to communicate groupings, connections and constraints. The obvious example of conveying meaning through layout in documents is tables.

Tables—textual tokens laid out in tabular form—are often used to compactly communicate information in fields (columns) and records (rows). They appear in the earliest writing on clay tablets, and in the most modern Web pages. Some make use of line-art, whereas others rely on white space only. They sometimes consist merely of two simple columns, other times of extremely baroque collections of headings, embedded subheadings, and varying cell sizes. They are used in everything from government reports, to magazine articles, to academic publications.

Given the importance of tables, we are interested in providing customized retrieval tools that deal specifically with this form of information. In our previous research (Pyreddy and Croft 1998), we developed heuristic algorithms for extracting tables from documents, and then indexed these tables as separate "pseudo-documents". A query to the system retrieved a ranked list of tables. Although this approach works reasonably well when tables are small and well-structured, there are problems in dealing with more realistic data, such as the huge amount of government data on the Web. In that case, tables are often very large and complex, and retrieving whole tables often does little to help users answer their information needs. In this paper, we instead adopt an approach based on question answering (QA), where the focus will be on extracting tables and then retrieving the specific cells of tables that address users' information needs. Question answering is a discipline of information retrieval (IR) that attempts to find specific answers in documents, relieving the users from having to scan retrieved documents for a desired information need. QA systems look for potential answer entities in a close relationship with query terms. However, in a table, the answer may be rows and columns away from the text that could contain the query terms such as the row names, column headers, titles and captions. Text tables, which can only be identified by their formatting in plain text, in particular make this difficult, since the layout of these tables varies enormously. Cell spacing is random. Some use characters (e.g., -,|,!,+) to delineate rows and columns. Columns may have multiple headers, or one header may cover a number of rows.

Another type of table found in web pages uses a markup language such as HTML or XML to designate the position of cells and presents a somewhat easier problem than text tables. The beginnings and ends of these tables are marked with tags, as are the column headers, individual cells and other items. However, there is no standard that requires an HTML/XML author to adhere to these. Information extraction from HTML/XML tables is not trivial, but instead presents a different set of problems. Tables, rows and cells are delineated by HTML/XML markups, so finding a table is not difficult. However, the table markups in HTML/XML are often used primarily to control layout rather than to interrelate content. It is often not clear if a "table" contains useful information. In addition, the use of markups is not consistent, and it is possible to embed plain text tables within table markups, so header lines and data cells still need to be identified, just as in text tables. It is our belief that the techniques presented here will also be applicable to HTML/XML tables. Since it is the harder problem, we focus on text table extraction in this paper.

Tables link their metadata (headers, titles) and cell data together. Cell data provide potential answers to information needs related to table content. In the approach described here, we create "cell documents" by automatically extracting table metadata and the associated cell data. The retrieval system indexes these cell documents and retrieves them in response to queries. Users will then be presented with a ranked list of specific table locations rather than a ranked list of long, complex tables.

Extracting and associating data and metadata requires a series of accurate decisions. For a typical table, the table itself must first be identified within the text document with data and header lines separated. Within the set of header rows, an algorithm must recognize the difference between titles and column headers and determine the span of each individual column header. Finally, row headers are identified. With this information in place, the data are combined to create a cell document for each table cell.

This paper presents models of table extraction and compares them using extraction accuracy and retrieval performance. The models integrate evidence from both content and layout by using heuristics and machine learning methods. In the heuristic method (Pinto et al. 2002), a Character Alignment Graph (CAG) is used to abstract a table to text characters and spaces and to extract individual cells for answer retrieval. As the method extracts cell data and associates them with their metadata, some language processing is used to distinguish headers from data rows. It was learned from this work that accurate tagging of tables is important in building the representation used for information retrieval. However, since this method did not provide a fine distinction between types of header rows, it tended to draw in extraneous metadata. Study of the retrieval errors shows that the extraneous metadata is a leading cause of failure.

The table extraction technique based on machine learning uses Conditional Random Fields (CRFs) (Lafferty et al. 2001) to label individual lines of text files. CRFs support the use of many rich and overlapping layout and language features, and as a result, they often perform significantly better than the heuristic method. Compared with the heuristic approach, the CRF model uses more discriminating labels and produces higher overall line labeling accuracy. Results are presented in Section 3. However, certain elements of tables (especially, headers) are more difficult to tag accurately. To solve the problem, new features and an increased training set are used to improve accuracy on these elements of tables. Better delineation of tables from text is also explored as means to improve retrieval performance. With these improvements, the final answer retrieval results on the Web documents containing government statistics data are 43%-100% higher than the heuristic method and 33%-57% higher than the original CRF extraction method. These improvements are also shown on a collection of Wall Street Journal articles.

The answer retrieval results from table extraction with the heuristic model, the original CRF extraction model and the improved CRF extraction model are presented in Section 4. Section 5 provides a discussion of these results, and Section 6 outlines some directions for future research.

2. OVERVIEW

2.1 Related Work

Question answering systems have many features in common (Voorhees 2000]). A question classifier is often used to determine an entity type (number, name, country, etc.) that is employed in the search for an answer. Typically, the next step is to retrieve documents or document passages that have a high likelihood of containing the answer. Those documents or passages are then searched for a short passage containing the answer. Finally, the actual answer string is extracted.

In previous work we created a system, QuASM (Pinto et al. 2002), applying the above procedure to table data. Part of this system involved the transformation of table data into cell documents that consist of data cells and related metadata. Based on the work of Pyreddy and Croft (Pyreddy and Croft 1997), a Character Alignment Graph (CAG) was employed to find text tables in documents. Heuristic algorithms then extract cell data and match them with their metadata. Retrieved cell documents were searched for answers based on finding named entities matching the class of a question. Our study of the errors made by the QuASM system shows that the extraneous metadata the system extracted is a leading cause of failure to retrieve the appropriate answers.

Hurst et al. (2000, 2001, 2000, 1995) describes the problem of information extraction from tables as one of both layout and language: the elements of tables are potentially ambiguous; cells may span multiple columns or multiple lines; header information may lay across multiple cells; there may be a lack of continuity between table lines. While systems can be based on either layout or language, the combination of the two is necessary to resolve the ambiguities in the data of tables. Given a table, Hurst's model breaks tabular text into blocks, and then determines what the blocks represent--using generative language models in both stages.

Ng, Lim and Koo (Ng et al. 2002) apply machine learning techniques to identify rows and columns in tables, but are only interested in finding the location of tables in text, not extracting its different components, either in terms of lines or cells.

2.2 Table Extraction and Answer Retrieval

The research described in this paper focuses on two main issues - table extraction and answer retrieval. Extraction transforms each data cell of a table into an individual cell document, consisting of the cell data and the metadata drawn from titles, headers, etc. Answer retrieval finds the answer from the cell documents created during table extraction. It ranks the cell documents using a language-modeling approach that is described in Section 4.

Table extraction has three key elements. The first is to locate a table in a document. The second is to determine the structures within the table, such as headers, footnotes, etc. The third is to associate the various elements of a table with their related data cells to create cell documents. An example of a text table can be seen in Figure 1a. Figure 1b shows an example of a processed table cell, with the column header information between the <COLUMN> tags, row header information between the <ROW> tags and title information between the <TITLE> tags. The cell document also includes the full table header lines (<CAPTIONS>) as context. This is helpful especially when the identification is not accurate although it does introduce some text that is not relevant for the cell, such as the full range of dates. In this example the column and row headers were extracted accurately, but for many of the cell documents some mistakes are made which result in text being incorrectly associated with cell data. It is our hypothesis that improvements in the extraction algorithm will reduce these mistakes and lead to improvements in answer retrieval.

2.3 Evaluation Methods

2.3.1 Data Sets

2.3.1.1 FedStats Data Set

A crawl of www.FedStats.gov performed in June 2001 gathered a large set of documents, many containing examples of text tables generated by government agencies. A heuristic is used to select a subset of these documents likely to contain tables. The data sets for experiments are chosen randomly from these documents, which might or might not contain tables. Each line of these documents is labeled by means of a simple heuristic program. Then, the machine labels were reviewed and corrected by human readers.

```
               Onions:  Area Planted and Harvested by Season, State,
                            and United States, 1996-98
-------------------------------------------------------------------------------
    Season     :          Area Planted          :         Area Harvested
     and       :----------------------------------------------------------------
    State      :   1996   :   1997   :   1998   :   1996   :   1997   :   1998
-------------------------------------------------------------------------------
               :                              Acres
               :
Spring         :
  AZ           :   2,100      2,100      2,500      1,900      2,100      2,500
  CA           :  10,100      9,900      7,000      9,600      9,600      6,800
  GA           :  16,000     16,200     15,000     14,700     15,800     13,900
  TX           :  15,300     12,400     12,000     13,000      9,800     11,400
  Total        :  43,500     40,600     36,500     39,200     37,300     34,600
```

(a) Original table

```
<QA_SECTION><QA_METADATA>
<TITLE> Onions: Area Planted and Harvested by Season State and United States 1996-98
</TITLE>
<CAPTIONS> Season Area Planted Area Harvested and State 1996 1997 1998 1996 1997 1998
Acres Spring </CAPTIONS>
<ROW> Spring AZ </ROW>
<COLUMN> Area Planted 1997 Acres </COLUMN>
</QA_METADATA>
2,100
</QA_SECTION>
```

(b) Extracted cell document corresponding to area planted, 1997, AZ

Figure 1. Text table example

We have four data sets on FedStats in total: a training set, a test set, a development set and an enlarged training set. The enlarged training set, which contains 255 more documents than the training set, is introduced to see if more training data can improve the system performance, as described in Section 3.1.3.2.2. Statistics of the data sets are given in Table 1.

Table 1. Statistics of data sets

Data Set	**# of Docs**	**# of Lines**	**# of Table Lines**
FedStats training set	52	31,915	5,764
FedStats test set	62	26,947	5,916
FedStats development set	6	5,817	3,607
FedStats enlarged training set	277	276,880	67,340
WSJ training set	98	5,683	1,401
WSJ test set	25	2,296	449

2.3.1.2 Wall Street Journal Data Set

The tables in the previously described FedStats data set are often complex, containing multi-level headers, super headers (spanning multiple columns), and a variety of sub-headers and section headers. Another data set, taken from the Wall Street Journal (WSJ) articles in the TREC (http://trec.nist.gov) collection, has a different style and contains mostly simple tables. We use the Wall Street Journal data set as a test set to validate our models and the improvements to them. Table 1 shows statistics of the training set and the test set we built on WSJ data set.

2.3.2 Table Extraction Evaluation

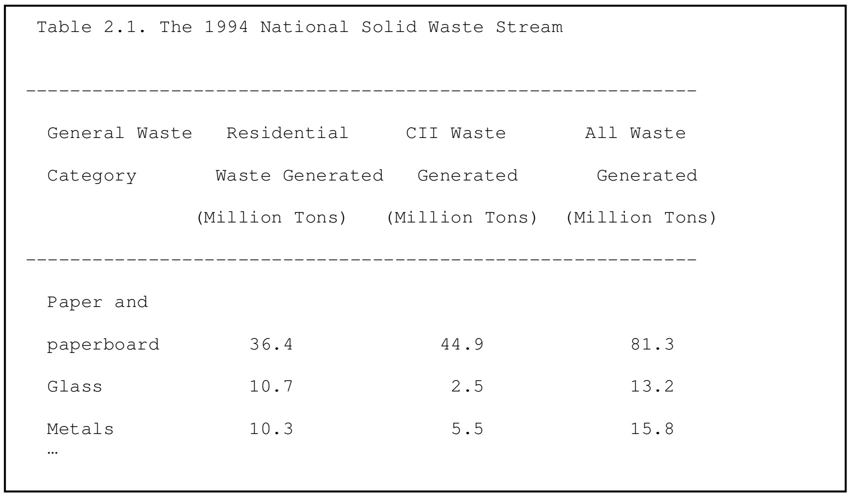
To evaluate the line labeling component of extraction, we use four measures: line accuracy, recall, precision and the F-measure. These four measures focus on different parts of line labeling performance. Line accuracy is the percentage of correct line labels over all lines. It shows the overall accuracy and general performance. Recall is the percentage of correct table lines the program labels over all table lines in the documents. Precision is the percentage of the correct table lines the program labels over all table labels. The F-measure is (2×Recall×Precision) ⁄ (Recall+Precision). Recall, precision and the F-measure show the system performance on a predefined set of labels for table components. A heuristic technique for labeling, which was developed by Pinto et al (Pinto et al. 2002), uses a smaller set of labels than the CRF-based technique we developed. In order to make the CRF results comparable with the heuristic results, multiple related CRF labels are combined to make one heuristic label.

To evaluate the cell association techniques, we assumed correct line labels and then made the association of metadata with cell data. The results of the associations were then compared to the correct associations and precision/recall for the associations were computed. In our experiments cell association heuristics work very well, as shown in Section 3.2, thus we have focused on the problem of line labeling in this discussion.

2.3.3 Answer Retrieval Evaluation

To evaluate the answer retrieval results, 50 questions for the tables in the FedStats data set and 53 questions for the tables in the Wall Street Journal data set were generated by hand. The questions were generated from the random set of documents that were selected from each data set. Each question asks about one cell in a table. For example, the answer to the question 'How many million tons of residential glass waste were generated in 1994?' corresponds to the cell containing '10.7' in the table in Figure 2. A sample list of questions is given in the appendix. This procedure for generating questions is somewhat unrealistic and can produce some complex questions that are unlikely to be found in a typical query log for Web search. Given more resources, it would be possible to develop a test collection with user-generated questions and relevance judgments, but it is important to realize that the typical user of such a search tool would be likely to be a "professional" searcher (such as government analysts, librarians, and paralegals) and would be more likely to ask longer and more complex queries. Given the artificial nature of this query set, the results should be viewed more as a test of extraction accuracy than of how the proposed retrieval model would perform in a realistic environment.

The content in the answer cell is used to check if the retrieval cell documents have the right answer. Then the Mean Reciprocal Rank (MRR) is computed for the first 1, 5 or 100 documents that are retrieved. The Reciprocal Rank of each individual query is the reciprocal of the rank at which the first correct response was returned in the first N responses or 0 if none of them contained a correct answer. The MRR score for a sequence of queries is the mean of the Reciprocal Rank values for each query in this set. MRR is often used as a question answering performance measure.



```
 Table 2.1. The 1994 National Solid Waste Stream


------------------------------------------------------------

  General Waste   Residential     CII Waste      All Waste

  Category       Waste Generated   Generated       Generated

                (Million Tons)   (Million Tons)  (Million Tons)
------------------------------------------------------------

  Paper and

  paperboard        36.4             44.9             81.3

  Glass             10.7              2.5             13.2

  Metals            10.3              5.5             15.8
  …
```

Figure 2. An example table used as a question source

3. TABLE EXTRACTION

3.1 Line Labeling

3.1.1 Line Labeling with CRFs

3.1.1.1 Conditional Random Fields

Conditional Random Fields (Lafferty et al. 2001) are undirected graphical models used to calculate the conditional probability of values on designated output nodes given values assigned to other designated input nodes.

In the special case in which the designated output nodes of the graphical model are linked by edges in a linear chain, CRFs make a first-order Markov independence assumption among output nodes, and thus correspond to finite state machines (FSMs). In this case, CRFs can be roughly understood as conditionally-trained hidden Markov models. CRFs of this type are a globally-normalized extension to Maximum Entropy Markov Models (MEMMs) (McCallum et al. 2000) that avoid the label-bias problem (Lafferty et al. 2001).

Let $o = (o_1, o_2, \ldots o_T)$ be some observed input data sequence, such as a sequence of lines of text in a document, (the values on *T* input nodes of the graphical model). Let *S* be a set of FSM states, each of which is associated with a label, $l_t \in L$, (such as a label DATAROW). Let $s = (s_1, s_2, \ldots s_T)$ be some sequence of states, (the values on *T* output nodes). CRFs with parameters $\Lambda = \{\lambda_1, \ldots\}$ define the conditional probability of a state sequence given an input sequence as

$$P_\Lambda(s \mid o) = \frac{1}{Z_o} \exp(\sum_t \sum_k \lambda_k f_k(s_{t-1}, s_t, o, t))$$

where *Zo* is a normalization factor over all state sequences (*o* is a observation sequence), $f_k(s_{t-1}, s_t, o, t)$ is an arbitrary feature function over its arguments, and λ_k is a learned weight for each feature function. A feature function may, for example, be defined to have value 0 in most cases, and have value 1 if and only if s_{t-1} is state #1 (which may have label Header), and s_t is state #2 (which may have label DATAROW), and the observation at position *t* in *o* is a line of text containing digits separated by more than one space. Higher λ weights make their corresponding FSM transitions more likely, so the weight λ_k in this example should be positive since widely-spaced digits often appear in data rows of tables. More generally, feature functions can ask powerfully arbitrary questions about the input sequence, including queries about previous lines, next lines, and conjunctions of all these. They may also have arbitrary values from -1 to 1.

CRFs define the conditional probability of a label sequence *l* based on total probability over the state sequences,

$$p_\Lambda(l \mid o) = \sum_{s:\ell(s)=l} p_\Lambda(s \mid o)$$

where $\ell(s)$ is the sequence of labels corresponding to the labels of the states in sequence *s*.

The normalization factor, *Zo*, (also known in statistical physics as the partition function) is the sum of the "scores" of all possible state sequences,

$$Z_o = \sum_{s \in S^T} \exp(\sum_{t=1}^{T} \sum_k \lambda_k f_k(s_{t-1}, s_t, o, t))$$

and the number of state sequences is exponential in the input sequence length, *T*. In arbitrarily-structured CRFs, calculating the partition function in closed form is intractable, and approximation methods such as Gibbs sampling, or loopy belief propagation must be used. In linear-chain structured CRFs (in use here for sequence modeling), the partition function can be calculated efficiently by dynamic programming (Byrd et al. 1994, Malouf 2002, Rabiner 1990, Voorhees 2000).

3.1.1.2 Line Labels

Our approach to table extraction starts by labeling each line of a document with a tag that describes that line's function in a table. The set of labels we use was designed by examining a large number of tables in Web documents. A good labeling accomplishes two goals: marking the boundaries of the tables (table location) and identifing the row types useful for answer retrieval and other applications. This section defines all the labels. An example of a text table with line labels is shown in Figure 3.

3.1.1.2.1 Nonextraction Labels

Non-extraction labels represent lines that do not contribute information about table cells. The three labels used in this paper are NONTABLE, BLANKLINE and SEPARATOR. NONTABLE represents lines of text that have no association with a table. NONTABLE lines usually appear outside of a table. BLANKLINE denotes lines that contain no visible text. BLANKLINE labels may appear within or outside a table. SEPARATOR indicates lines that use certain punctuation characters (-,*, e.g.) to suggest sectioning. They can appear anywhere in a document.

```
<TITLE>                   Onions:  Area Planted and Harvested by Season, State,
<TITLE>                            and United States, 1996-98
<SEPARATOR>     ------------------------------------------------------------------------------
<SUPERHEADER>       Season   :        Area Planted         :        Area Harvested
<SUPERHEADER>         and    :----------------------------------------------------------------
<TABLEHEADER>        State   :  1996  :  1997  :  1998  :  1996  :  1997  :  1998
<SEPARATOR>     ------------------------------------------------------------------------------
<SUBHEADER>                  :                         Acres
<SEPARATOR>                  :
<SECTIONHEADER> Spring       :
<SECTIONDATAROW>  AZ         :  2,100     2,100     2,500     1,900     2,100     2,500
<SECTIONDATAROW>  CA         : 10,100     9,900     7,000     9,600     9,600     6,800
<SECTIONDATAROW>  GA         : 16,000    16,200    15,000    14,700    15,800    13,900
<SECTIONDATAROW>  TX         : 15,300    12,400    12,000    13,000     9,800    11,400
<SECTIONDATAROW>  Total      : 43,500    40,600    36,500    39,200    37,300    34,600
```

Figure 3. Example text table with line labels

3.1.1.2.2 Header Labels

Header labels mark lines that contain metadata for data cells. Some or all of the information in header lines will be associated with the data cells below it. The header labels include TITLE, SUPERHEADER, TABLEHEADER, SUBHEADER and SECTIONHEADER. TITLE represents lines of text in which all content should be associated with every data cell in the table. SUPERHEADER lines contain text whose association with data cells spans multiple columns. SUPERHEADER lines appear above TABLEHEADER lines. TABLEHEADER represents lines where a one to one correspondence between data columns and header cells is likely. SUBHEADER, like SUPERHEADER, has text with multiple column associations, but appears below the TABLEHEADER label. SECTIONHEADER indicates lines of text that pertain to the next few lines of data. SECTIONHEADER labels often group together data lines that are sub-topics of the SECTIONHEADER.

3.1.1.2.3 Data Row Labels

Data row labels mark rows that contain data cells. Data rows also often contain header information for the rows. The data row labels include DATAROW and SECTIONDATAROW. DATAROW represents lines whose column headers are found in SUPERHEADER, TABLEHEADER and SUBHEADER lines. SECTIONDATAROW represents lines whose data cells are also headed by SECTIONHEADERS.

3.1.1.2.4 Caption Labels

Caption labels mark rows that appear below data but still apply to the table. The caption labels include TABLEFOOTNOTE and TABLECAPTION. TABLEFOOTNOTE represents a reference to a cell or line in a table. TABLECAPTION represents a line of text that refers to the whole table.

3.1.1.3 Feature Set

The feature set we use was designed not only by examining a large number of tables in Web documents, as we did with the label set, but also by experimenting on development data. Good features can help improve the line labeling performance. As the system was developed, features were added or removed to improve performance on development data. This section explains these features. All features are binary valued except the ones that are indicated to be expressed in percentages. The features we used in this paper are not independent, and sometimes overlap. The CRFs are good at handling arbitrary overlapping features (Lafferty et al. 2001).

3.1.1.3.1 White Space Features

White space is employed in documents and tables to improve readability. Common uses are to separate table cells, indent titles, and indent sub-section data rows and to provide a separation between lines of text. For purposes of this paper, white space is any character matching the regular expression "\s" as defined in the Java Pattern class. The white space features:

- *Four consecutive white space characters*: this feature is often found in data rows, separating row headers from data, or in titles that are centered.
- *Four space indents*: often found in title lines, or in header lines where the row header column is not labeled, or at the start of sub-headers and super-headers.
- *Two gaps*: at least two consecutive white spaces between non-space characters are gaps. Often used to separate cells in data and header lines. Containing at least two gaps is used as an indicator of a table line.
- *Large gap:* at least five consecutive white spaces is a large gap. Sometimes used in tables with a small number of columns to separate a row header from data.
- *Single space indent:* often found in section data rows, as the indent sets these rows off from normal data rows.
- *All space characters:* the feature of a line that would match the regular expression "^\s*$", a blank line.
- P*ercentage of white space:* often separates data rows from text. Data rows tend to have a higher percentage of white space.

3.1.1.3.2 Text Features

Printable characters also convey information about the type of line being observed. The use of digits, keywords and the layout of lines all contribute features that make lines recognizable as being part of a table or not. The text features include:

- *Three cells on a line*: this is a common feature of data lines.
- *Header features*: strings common in table headers (month abbreviations, year strings (e.g., 1981), and other key words) constitute header features. This feature is expressed as a percentage of the characters in a line found in such strings.
- *Alphabet characters* (A-Za-z): this feature is expressed as a percentage of the non-white-space characters in the line. This is useful in distinguishing numeric data rows from text headers.
- *Digit characters* (0-9): this feature is also expressed as a percentage of the non-white-space characters in the line. It is also useful in distinguishing numeric data rows from text headers in tables.

3.1.1.3.3 Separator Features

Punctuation marks are often used for formatting tables. A line of dashes may delineate the header section from the data section of a table. Vertical characters mark the boundaries between cells. Two features are based directly on punctuations:

- *Separator characters* (-,+,—,!,=,:,*): this feature is expressed as a percentage of the non-space characters in the line. It is often used to delineate the boundaries between sections of tables (headers from data) and to mark column boundaries.
- *Four consecutive periods* (.): indicative of tables where the row header is separated by a large distance from a single data column, such as the ones in a table of contents. The periods may or may not be separated by white space characters.

3.1.1.3.4 Feature Representation

Each feature can be represented as a binary value. A 1 indicates the presence of a feature and a 0 indicates the lack of a feature. For percentage features, a threshold is set, above which the feature will have the value 1, otherwise 0. Thresholds used in this paper are listed in Table 2. With CRFs, it is as easy to use continuous-valued features as it is to use discrete-valued ones. This enables experiments in which the percentage features are not discretized as binary features, but the actual percentages are used instead.

Table 2. Thresholds for percentage features

Feature	**Threshold**
Percentage of white space	30%
Header features	60%
Alphabet characters	60%
Digit characters	70%
Separator characters	80%

3.1.1.3.5 Conjunctions of Features

CRFs have the ability to take into account information from before and after the current label. One way of accomplishing this is to look at a conjunction of features. The values of features on one line are multiplied by the values of features on another (or the same) line, creating a new feature (e.g., Feature1&Feature2). Conjunctions help capture relationships that a linear combination of features may not. In these tests, the conjunctions used are the current line with the previous line, the current line with the following line and the conjunction of the two following lines together.

The feature set we used in this paper is designed based on our development set and the tables we examined. We believe that similar features can be applied to other corpora. Some features, like the four consecutive white space characters, four consecutive periods and large gap, may need to be changed if the document and table style in the new corpus is very different. The thresholds in these features are quite flexible. Also, a couple of features (e.g., alphabet/digit characters) may not be so effective on a corpus containing few numeric data cells. Therefore, when designing a new feature set for another corpus constructing a new development data set from that corpus will probably be helpful, and sometimes necessary.

3.1.2 Line labeling with Heuristics

As a baseline and comparison for the CRF extractor, we also used a heuristic extractor developed by Pinto et al. (2002). This heuristic method uses many of the same features as the CRF extractor. The first step of Pinto's method is to create a Character Alignment Graph (CAG) to look for white space alignment in blocks of contiguous lines of texts. A number of heuristics are then applied to the CAG, which include:

- A row with more than two gaps may be a table row. Gaps are large areas of white space in a row, and may indicate column structure.
- A row with more than four consecutive white spaces may be a table row.
- The density of table rows (number of rows marked as belonging to a table) indicates the presence of a table.
- Simple rows (with less than 3 cells) can indicate the beginning or end of a table.
- Rows at the beginning of a table are likely to be less regular than the rest of the table.
- The number of simple rows in a table should be small.
- Non-table rows and blank rows are a sign of the end of a table.
- A number of consecutive signs of the end of a table will indicate the end of a table more strongly.

A number of additional heuristics are used to extract table headers. Headers are especially difficult to detect since they can span multiple lines, be split from the data by lines containing other information, and can have both sub-headers and super-headers. The additional heuristics include:

- Simple rows at the beginning of a table are probably titles.
- Other rows at the beginning of a table are probably caption rows.
- Simple rows in the middle of a table are likely caption rows.
- Rows in a table with a similar number of cells are likely data rows.
- The most common number of cells in a row represents the number of cells in a data row, and rows with that number of cells are marked as such. Authors of text tables often leave cells without data blank. This makes these rows to appear to have fewer cells.
- Data rows with integer numbers in the range 1700 to 2100 are likely caption rows, since these numbers probably represent years. (These are likely year headers for US government studies, given the age of the country.)
- When data rows are made up of cells containing digits, data rows made up of alphabetic characters are probably caption rows.
- If a cell in a caption row can cover multiple cells in a data row, it is included in the headers for all those columns.
- Caption rows in the middle of a table indicate a new section of the table, and the headers from that row should be only applied to the next section.

- Cells in the first column are used as row headers.

The results presented in (Pinto et al. 2002) showed that this table extraction technique performed better than previous heuristic methods (Pyreddy and Croft 1997), and was adequate to support a reasonable level of question answering performance.

3.1.3 Experiments

3.1.3.1 Experimental Results

We implemented our CRF labeler using Mallet (McCallum 2002) – a machine learning for language toolkit. The table extraction results from the CRF and the heuristic labeler using FedStats data and WSJ data are shown in Table 3 and Table 4, respecitively. Header lines here represent all the header labels in Section 3.1.1.2.2, and table lines represent all labels except the nonextraction labels in Section 3.1.1.2.1.

From these results we can see that for FedStats, the heuristic method works better than the CRF model on header line recall, although CRF labeling has higher line accuracy, precision, recall and the F-measure for table lines. For WSJ data, the performance of the CRF labeler is significantly better than the heuristic labeler, but there is still room for improvement. It is useful to examine where the CRF labeler made errors, especially for the FedStats data. Table 5 presents recall and precision for each of the labels for the CRF run on the FedStats test data. In this data set, labeling of TABLEHEADER lines was quite poor and, even worse, the majority of mislabeled TABLEHEADER lines were labeled as NONTABLE or DATAROW. NONTABLE labels are the most frequent mistake. SECTIONDATAROW lines were most often mislabeled as DATAROW lines, although the DATAROW lines were most often missed when the table was missed. A number of NONTABLE lines were mislabeled as TABLECAPTION, and these errors did not occur in the context of the end of a table. This aspect of the CRF needs to be further explored.

Further analysis of the errors indicated that the main problem is the low recall of header line labeling by the CRF model, as shown in Tables 3, 4 and 5. Improving the performance for this class of table lines is crucial, since they provide the link that connects table cells with a query.

Table 3. CRF line labeling performance on FedStats

	Line Accuracy		**Precision**	**Recall**	**F-measure**
CRF	93.5%	Table lines	96.9%	83.9%	0.899
		Header lines	50.4%	57.4%	0.537
Heuristic	80.4%	Table lines	56.1%	75.8%	0.645
		Header lines	11.1%	67.9%	0.191

Table 4. CRF line labeling performance on WSJ

	Line Accuracy		**Precision**	**Recall**	**F-measure**
CRF	95.6%	Table lines	97.8%	79.1%	0.875
		Header lines	45.8%	73.2%	0.563
Heuristic	85.3%	Table lines	58.6%	69.0%	0.634
		Header lines	14.8%	61.4%	0.239

Table 5. Recall/Precision of each label from CRF model, FedStats

Label	# of Labels	Precision	Recall
NONTABLE	14118	.954	.979
BLANKLINE	6186	.998	.993
SEPARATOR	492	.942	.896
TITLE	192	.897	.542
SUPERHEADER	92	.909	.652
TABLEHEADER	236	.340	.462
SUBHEADER	49	.616	.918
SECTIONHEADER	186	.701	.441
DATAROW	4497	.912	.864
SECTIONDATAROW	716	.678	.547
TABLEFOOTNOTE	163	.896	.687
TABLECAPTION	20	.000	.000

The example in Figure 4 shows how the CRF labeler missed the headers. The labels at the beginning of the lines are the labels that the CRF labeler produced for the following text. For the table in this figure, the labeler mislabeled the beginning part of the table, including the title and the header, as <NONTABLE> and <DATAROW>. These lines do not contain the features which are indicative of titles and headers for the CRF labeling model. The header is missing separator characters that usually indicate column names, and leading white spaces are absent from the title. Tables like this led us to develop new features to better capture this type of layout.

```
<BLANKLINE>
<NONTABLE> Table 2.1. The 1994 National Solid Waste Stream
<BLANKLINE>
<BLANKLINE>
<SEPARATOR>--------------------------------------------------------------
<BLANKLINE>
<DATAROW>  General Waste    Residential     CII Waste        All Waste
<BLANKLINE>
<DATAROW>  Category        Waste Generated   Generated        Generated
<BLANKLINE>
<DATAROW>                (Million Tons)   (Million Tons)   (Million Tons)
<BLANKLINE>
<SEPARATOR>--------------------------------------------------------------
<BLANKLINE>
<DATAROW>  Paper and
<BLANKLINE>
<DATAROW>  paperboard         36.4             44.9              81.3
<BLANKLINE>
<DATAROW>  Glass              10.7              2.5              13.2
<BLANKLINE>
<DATAROW>  Metals             10.3              5.5              15.8
…
```

Figure 4. An example of mislabeled headers

3.1.3.2 Improvements

Based on the error analysis in Section 3.1.3.1, we try to improve the line labeling performance, especially for header lines, by adding new features and increasing the training set for FedStats data.

3.1.3.2.1 New Features

In order to improve the CRF labeling results, we add new features. A study of the errors made with the current set of features led to the creation of two new feature groups. One describes the layout of characters, especially the alignment of the current line with its neighboring lines. When a person looks at a table, there is clear overlap between the space and non-space characters in data rows and headers. Measurements of this overlap are integrated as new features. Another group of features describes the language used in titles and headers. Our observation is that certain key words appear frequently in certain lines. For example, the title may contain 'Table'.

We develope a number of other features and tested them individually. We select the top 20 features that have the best results and are representative, and incorporate these features into the new CRF labeler. Table 6 lists some of the new features and the percentage improvement obtained with those features.

Table 6. New features with their improvements

Features	Improvements
Percentage of overlapping blank/non-blank characters of a line with the *n*th above/next non-blank-lines.	7% on recall
Among *n* lines around current line $\sum_i \lvert x_i - \bar{x} \rvert$, *i=1..n, x* is the number of gaps	3% on recall
Percentage over all gaps of strict overlapping gaps of this line with the *n*th above/next non-blank-lines. Strict overlapping gaps are the overlapping gaps sharing close boudaries.	8% on recall
Percentage of words beginning with capital letters	4% on recall
Containing 'table' or 'figure' at the beginning of a line	7% on recall and 5% on precision
Containing formatting tags "< ... >"	5% on recall

3.1.3.2.2 Increased Training Data

In addition to the new features, we also explore the effect of the size of training set. To test the hypothesis that more training data would lead to better header labeling, a ten-fold cross-validation experiment of FedStats data was performed. The data in the test set and the original training set are pooled and split into ten similar size parts, each containing roughly the same number of documents. We test the CRF labeling performance for header lines on these ten folds respectively with the other nine as a training set. The results are obtained without the new features detailed in Section 3.1.3.2.1 and are shown in Table 7.

Table 7. Cross-validation experiments on FedStats, header accuracy

Test set	Original CRF			Heuristic		
	Precision	Recall	F-measure	Precision	Recall	F-measure
1	100%	81%	0.90	0.8%	69%	0.02
2	96%	88%	0.92	10%	70%	0.18
3	95%	99.9%	0.97	56%	80%	0.66
4	99%	98%	0.98	61%	80%	0.69
5	96%	66%	0.78	46%	98%	0.63
6	92%	86%	0.89	41%	89%	0.56
7	66%	17%	0.27	9%	44%	0.15
8	80%	92%	0.86	2%	77%	0.04
9	93%	59%	0.72	15%	78%	0.25
10	94%	89%	0.91	31%	75%	0.44
Overall	95%	86%	0.90	26%	78%	0.39

Most of the results in Table 7 are considerably better than the counterparts in Table 3 and Table 4. Compared with the previous experiments, the major difference is that the size of training set is relatively much larger compared with the size of test set. This suggests that increasing the size of training set will improve the recall and precision on headers. Therefore, we add another 225 documents into the FedStats training set and have the enlarged training set described in Section 2.3.1.1. In order to show the relation between the size of training set and table extraction performance, we split this increased training set into four similar-size subsets and did four tests, with one to four subsets as training set.

The results are shown in Table 8. The new features in Section 3.1.3.2.1 are incorporated in the experiments. From these results we can see that the recall, especially the crucial header line recall, improves when the training set increases.

Table 8. CRF table extraction performance by the size of training data set, FedStats

Training Data Sets (sum-size)	Line Accuracy		Precision	Recall	F-measure
1 (5.6M)	90.9%	Table lines	98.0%	75.1%	0.850
		Header lines	59.7%	42.7%	0.498
2 (11.2M)	91.2%	Table lines	91.6%	81.4%	0.862
		Header lines	78.9%	55.3%	0.650
3 (16.8M)	90.2%	Table lines	97.3%	83.5%	0.899
		Header lines	67.2%	62.0%	0.645
4 (22.5M)	92.2%	Table lines	97.2%	84.3%	0.903
		Header lines	62.9%	65.4%	0.641

3.1.3.3 Results after Improvements

New results from FedStats data after the improvements described in Section 3.1.3.2 are actually shown in the bottom line of Table 8. As a comparison they are shown again in Table 9 together with the original CRF results and heuristic results from Table 3.

Table 9. CRF table extraction performance on FedStats

	Line Accuracy		**Precision**	**Recall**	**F-measure**
Heuristic	80.4%	Table lines	56.1%	75.8%	0.645
		Header lines	11.1%	67.9%	0.191
Original CRF	93.5%	Table lines	96.9%	83.9%	0.899
		Header lines	50.4%	57.4%	0.537
Improved CRF	92.2%	Table lines	97.2%	84.3%	0.903
		Header lines	62.9%	65.4%	0.641

Table 9 shows that, while the improved CRF extraction does not do as well on line accuracy as the original, it does much better on header lines. While it does not capture as many headers as the heuristic method, the superior precision of the CRF method results in less extraneous metadata. As we mentioned before, header line labeling is especially important and our main object for improvements. The improved CRF is the one that achieves the best overall performance on header lines. It also achieves the best F-measure.

Table 10 shows the results on the WSJ data set after the new features are added. (We do not have an enlarged training set on WSJ.) The improved CRF extraction does much better on all measures.

Table 10. CRF table extraction performance on WSJ

	Line Accuracy		**Precision**	**Recall**	**F-measure**
Heuristic	85.3%	Table lines	58.6%	69.0%	0.634
		Header lines	14.8%	61.4%	0.239
Original CRF	95.6%	Table lines	97.8%	79.1%	0.875
		Header lines	45.8%	73.2%	0.563
Improved CRF	97.8%	Table lines	98.3%	91.3%	0.947
		Header lines	74.7%	82.9%	0.786

3.1.3.4 Table Delimitation

With the label set described in Section 3.1.1.2, tables are delimited by a certain set of labels, like NONTABLE, TITLE or TABLEHEADER. Unfortunately, at our level of header recall, we are still mislabeling quite a number of header lines. The algorithm we used determines the boundaries of tables by the labels, which is weak at robustness of delimination. For example, some TABLEHEADER lines in

Figure 4 were labeled correctly after the CRF labeler was improved. The new labels are shown in Figure 5. However, the grey line labeled <NONTABLE> caused the extraction program to miss the headers of this table.

We changed the delimitation rules in the program and introduced ‘tolerance’ into them to alleviate the missing-header problem . The main flow of the rules is the following:

- When the system encounters a super header line, like <TITLE>, <TABLEHEADER>, or <SUPERHEADER>, if the tolerance is 0 then give the tolerance a value, which is 7 in our current system; if the tolerance is more than 0 then the table ends in the above line.
- When the system encounters a sub header line, like <SUBHEADER> or <SECTIONHEADER>, give the tolerance a smaller value. We use 5 in our program.
- When the system gets a <BLANKLINE> or a <NONTABLE>, reduce the tolerance.
- When the tolerance gets 0, the table ends.

The improvements obtained by this procedure are shown in the answer retrieval experiments in Section 4.2.

```
<BLANKLINE>
<NONTABLE>    Table 2.1. The 1994 National Solid Waste Stream
<BLANKLINE>
<BLANKLINE>
<SEPARATOR>    ------------------------------------------------------------
<BLANKLINE>
<TABLEHEADER> General Waste   Residential     CII Waste       All Waste
<BLANKLINE>
<TABLEHEADER> Category       Waste Generated   Generated       Generated
<BLANKLINE>
<NONTABLE>                  (Million Tons)   (Million Tons)  (Million Tons)
<BLANKLINE>
<SEPARATOR>    ------------------------------------------------------------
<BLANKLINE>
<DATAROW>      Paper and
<BLANKLINE>
<DATAROW>      paperboard        36.4             44.9             81.3
<BLANKLINE>
<DATAROW>      Glass             10.7              2.5             13.2
<BLANKLINE>
<DATAROW>      Metals            10.3              5.5             15.8
…
```

Figure 5. An example of missed headers caused by <NONTABLE>

3.2 Cell Association

The second step in extracting table data is to match data cells with their appropriate headers. A number of heuristics were used to accomplish this:

- If a cell in a header row can cover multiple cells in a data row, it is included in the headers for all covered columns.
- Header rows in the middle of a table indicate a new section of the table, and the headers from that row should only apply to the next section.
- Cells in the first column are used as row headers.

One of the most important parts of the above heuristics was to determine if a cell in a caption row covered multiple cells in a data row. The most common number of cells in a line is calculated by creating a histogram of the cells in each line and picking the most frequent entry. For a line with this number of cells, the character positions where each cell begins and ends are recorded. Then for a caption line with fewer cells, the same information are recorded. Any columns that fall under the span of the caption row cell get this cell as a piece of its metadata. In this way most column headers can be captured perfectly except in rare cases, e.g., the cell delimitation in some complicated and compact tables may not be neat enough. To test this, we selected 62 tables that have column headers and table rows, and checked to see if we were able to capture the column header information by the histogram of column positions. The results are in Table 11.

Table 11. Header capture experiment

Number of Columns	425
Columns, Header Text Missing	27
Columns, All Header Text Captured	398
Percent, All Header Text Captured	93.6

The above test shows the performance for capturing columns. We also evaluated cell association by header cells. We selected 115 tables and let the system associate each cell with its relevant information based on correct line labels. Then we computed the precision and recall of the header items that were associated correctly. The results are shown in Table 12. From the results in these two tables we can see that the cell association heuristics work well.

Table 12. Cell association experiment

	Column Headers	Row Headers
Recall	99.4%	95.7%
Precision	92.5%	99.9%

4. ANSWER RETRIEVAL

4.1 Answer Retrieval with Language Models

Answer retrieval attempts to find the cell documents that contain the answers to queries. The cells were put into cell documents with metadata, and can be extracted as answers when the documents are retrieved. Given a query, cell documents are ranked using a language model framework for retrieval (Ponte and Croft 1998). We expect that the improvements to the extraction algorithm described in Section 3 will lead to improvements in answer retrieval as well.

4.1.1 Database Building

The database of cell documents was indexed using Lemur (Ogilvie and Callan 2002) – a language-modeling and information retrieval toolkit. Databases were built both with and without stemming and stopping to see how these variations affect performance. We used the Krovetz stemmer (Krovetz 1993) and the default stopword list of Lemur.

4.1.2 Answer Retrieval

The cell documents were ranked using language-modeling techniques (Berger and Lafferty 1999, Ponte and Croft 1998). The basic approach of using language models for IR assumes that the user generates a query as text that is representative of the "ideal" document. The task of the system is then to estimate, for each of the documents in the database, which is most likely to be the ideal document. That is, we calculate:

$$\arg\max_{D} P(D \mid Q) = \arg\max_{D} P(Q \mid D) P(D)$$

where the prior $P(D)$ is usually assumed to be uniform and a language model $P(Q \mid D)$ is estimated for every document. The baseline for the query-likelihood model is this equation, assuming uniform prior probabilities. In other words, we rank cell documents by

$$P(D \mid Q) \propto \prod_{i=1}^{n} P(q_i \mid D)$$

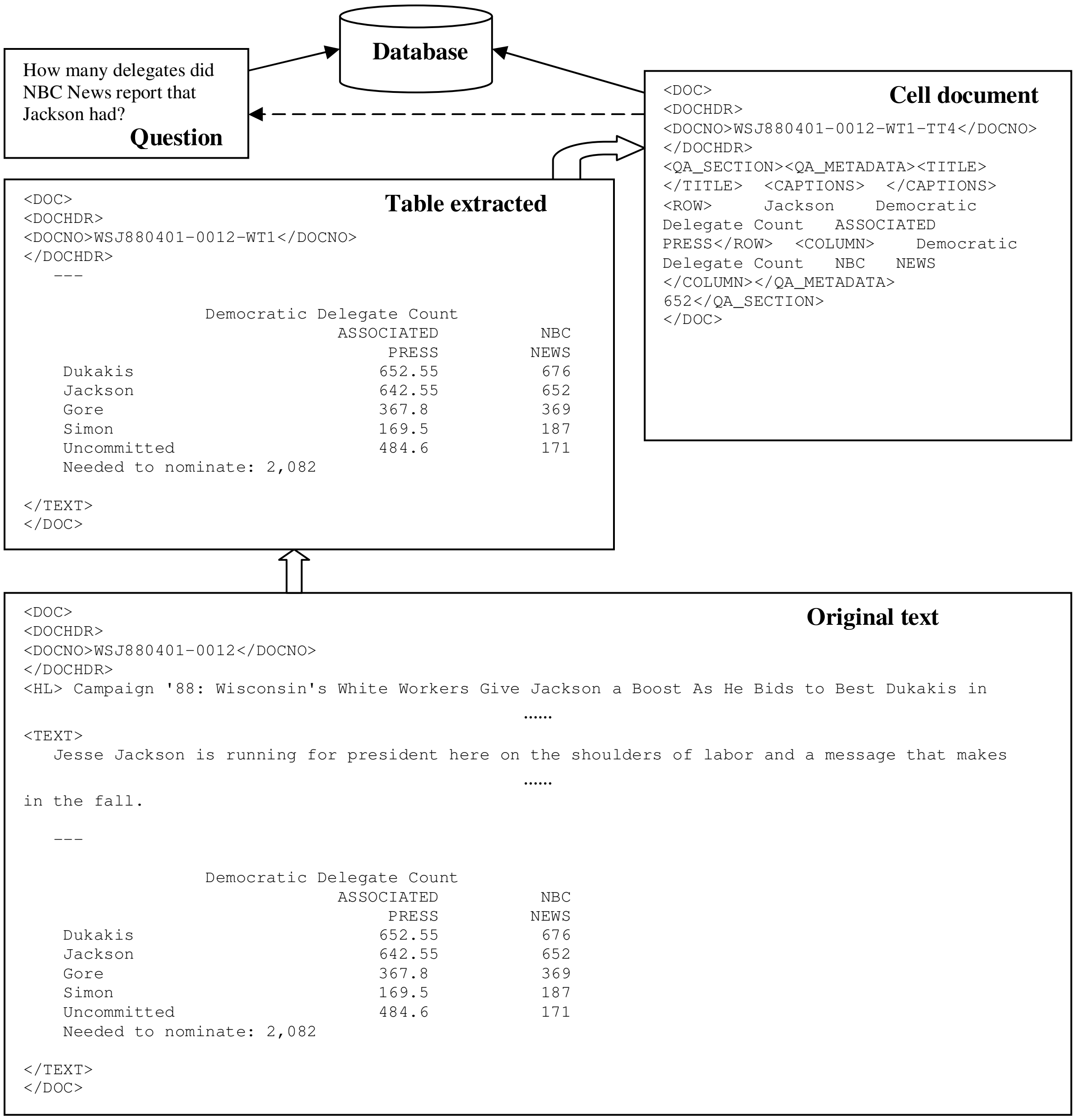
where D is a cell document, Q is a query and q_i is a query term in Q. We used interpolation with a collection model to smooth probabilities (Zhai and Lafferty 2001). The collection probabilities were estimated using the entire collection.

Figure 6 gives an example of the complete procedure of answer retrieval for the question "How many democratic delegates did NBC News report that Jackson had?". The original text is from the Wall Street Journal data set. The table was extracted by the CRF extractor and cell documents were constructed. The cell document in Figure 6 was retrieved and it contains the answer.

4.2 Experimental Results

This section presents answer retrieval results on the extracted data (cell documents) by the table extraction methods discussed in Section 3. Answer retrieval effectiveness depends heavily on the quality of table extraction, so these experiments should be viewed primarily as an evaluation of the extraction methods and the method of generating the cell documents.

We tested the improvements described in Section 3.1.3.2 on the whole FedStats data set in two increments. The first step (called "modified" in the tables) tested the new CRF labeler trained with the new features and the enlarged training set; the second step (called "final") added the algorithm that is more tolerant of non-table lines in the headers. Table 13 presents the results of these experiments, along with the results from the heuristic extraction (extraction with heuristic labler) and the results from the original CRF extraction (extraction with CRF labler) for comparison (The same improvements were not applied to the heuristic extraction as to the original CRF extraction because there is no training set in the heuristic extraction and the heuristic labeling rules (Pinto et al. 2002) were already finely tuned for robust table delimitation). Table 14 shows the improvements in MRR in percentage terms. Results are shown for both the improvement over the heuristic extraction and the original CRF extraction.



How many delegates did NBC News report that Jackson had?

```
<DOC>
<DOCHDR>
<DOCNO>WSJ880401-0012-WT1-TT4</DOCNO>
</DOCHDR>
<QA_SECTION><QA_METADATA><TITLE>
</TITLE>  <CAPTIONS>  </CAPTIONS>
<ROW>     Jackson    Democratic
Delegate Count   ASSOCIATED
PRESS</ROW>  <COLUMN>    Democratic
Delegate Count   NBC   NEWS
</COLUMN></QA_METADATA>
652</QA_SECTION>
</DOC>
```

```
<DOC>
<DOCHDR>
<DOCNO>WSJ880401-0012-WT1</DOCNO>
</DOCHDR>
   ---

                  Democratic Delegate Count
                               ASSOCIATED          NBC
                                    PRESS         NEWS
    Dukakis                        652.55          676
    Jackson                        642.55          652
    Gore                           367.8           369
    Simon                          169.5           187
    Uncommitted                    484.6           171
    Needed to nominate: 2,082

</TEXT>
</DOC>
```

```
<DOC>
<DOCHDR>
<DOCNO>WSJ880401-0012</DOCNO>
</DOCHDR>
<HL> Campaign '88: Wisconsin's White Workers Give Jackson a Boost As He Bids to Best Dukakis in
                                                         ......
<TEXT>
   Jesse Jackson is running for president here on the shoulders of labor and a message that makes
                                                         ......
in the fall.

   ---

                  Democratic Delegate Count
                               ASSOCIATED          NBC
                                    PRESS         NEWS
    Dukakis                        652.55          676
    Jackson                        642.55          652
    Gore                           367.8           369
    Simon                          169.5           187
    Uncommitted                    484.6           171
    Needed to nominate: 2,082

</TEXT>
</DOC>
```

Figure 6. An example of answer retrieval

Table 13. MRR for answer retrieval on FedStats

	MRR at	Heuristic	Original CRF	Modified	Final
No stemming/ stopping	1	0.14	0.14	0.18	0.22
	5	0.177	0.171	0.2	0.256
	100	0.187	0.177	0.212	0.268
With stemming/ stopping	1	0.12	0.18	0.24	0.24
	5	0.177	0.221	0.284	0.306
	100	0.199	0.234	0.297	0.322

Table 14. Percentage Improvements on FedStats

	MRR at	vs. Heuristic		vs. Original CRF	
		Modified	Final	Modified	Final
No stemming/ stopping	1	29%	57%	29%	57%
	5	13%	45%	17%	50%
	100	20%	43%	17%	51%
With stemming/ stopping	1	100%	100%	33%	33%
	5	60%	73%	29%	38%
	100	49%	62%	27%	38%

The results for answer retrieval on the whole WSJ data set are presented in Table 15. The 2-step improvement settings are the same as above except the training set was not enlarged. The answer retrieval results from the simpler WSJ tables extracted by the CRF extractor are also improved considerably by the same ideas that worked for the more complex tables in FedStats. In addition, the results from the WSJ tables are much better than FedStats tables. The WSJ tables are editorially different than the FedStats tables. Titles in these tables are often not included, or are formatted in such a way that they appear as the last paragraph of a story. Answer retrieval results are considerably better on these simple tables because the extraction quality is higher, as shown in Table 10.

Table 15. MRR for QA retrieval on WSJ

	MRR at	Heuristic	Original CRF	Final
No stemming/ stopping	1	0.073	0.377	0.415
	5	0.098	0.437	0.488
	100	0.111	0.450	0.503
With stemming/ stopping	1	0.073	0.415	0.434
	5	0.094	0.458	0.491
	100	0.103	0.474	0.506

5. DISCUSSION

In this paper we present and evaluate approaches to table extraction for answer retrieval, and develop a number of improvements for the CRF extraction model. The CRF labler shows excellent line accuracy, but this does not always translate to high effectiveness in retrieving answers. The error analysis leads to a number of improvements that directly impacted retrieval performance.

5.1 CRF vs. Heuristic

Initially, the answer retrieval results from the cell documents created by the heuristic extraction were better than the results from the cell documents created by the CRF extraction. Although the CRF model has significantly better accuracy labeling lines, it does not identify as many headers as the heuristic. The cross-validation experiments show that with more training data, CRF labeling performance on header lines improves considerably. After incorporating various improvements, the cell documents generated by the CRF extractor are superior for answer retrieval.

The experiments with the WSJ data set indicate that the heuristic approach is stable across many types of tables. However, header line precision is poor, and the heuristics do not distinguish enough between types of header rows, which limits its overall retrieval performance. In contrast, the CRF extraction is more adaptable. Its retrieval results can be improved considerably by the methods we adopted in these experiments. In particular, increasing the size of training set has a positive effect on header labeling, and the inclusion of more varied features also leads to increased performance.

5.2 QA Retrieval Error Analysis

Errors in answer retrieval can be caused by either the extraction technique or the retrieval model. For a particular query, the QA system may fail in retrieving a document containing the answer, or a document with the correct answer may be ranked very low by the answer retrieval algorithm. By classifying and examining the errors, further improvements can be achieved.

This analysis supports earlier observations that accurate table extraction is crucial for good retrieval performance. The line accuracy, precision and recall of table lines and header lines are all very important, especially header line recall, which shows a dominant role in the experiments. The question, "How many thousands of pounds of carpet class fibers were consumed in spinning in 1991?" is an example of how an answer can be missed by poor extraction. In the table containing the answer, the keywords for this query, including "fibers", "1991", "carpet class" and "thousands of pounds" are in the headers of the corresponding table. The extraction program, however, mislabeled all the header lines. Then all the column headers and section headers were missed in the cell document for the answer cell. Only a "carpet class' was extracted with the answer cell number as a row header. The cell document containing the answer was not ranked in the top 100 retrieved documents, thus the answer was totally missed. This points out the need to continue working on improving CRF extraction model, especially to increase the recall of header lines to solve such problems.

The retrieval model is also important for the system performance. The question, "What was the bearing acreage for tart cherries in 1995?" is an example of how a cell document can be poorly ranked by the answer retrieval algorithm. The cell was correctly extracted from its table. There is, however, more than one table on this question topic in the dataset. Some of these tables present the bearing acreage for tart cherries for US states. The table containing the answer also has the numbers listed with state numbers, but has a total line at the end. The tables were extracted correctly, but the current retrieval methods failed in locating the answer due to the lack of the query term "total" in the question. The cell documents for other cells that are in similar tables contain all the keywords of this question, but the correct document contains the word 'Total' instead of a state name. Since an important query term is missing, the correct document is given a low rank on this question.

Other improvements may require increased coordination between the table extraction and answer retrieval algorithms. For example, an answer retrieval algorithm that puts more weight on row and column metadata may produce better rankings.

6. CONCLUSIONS AND FUTURE WORK

We have shown that it is possible to identify tables and retrieve answers from those tables at a reasonable level of accuracy. The heuristic extractor is consistent across databases but its performance is poor. The CRF extractor performs significantly better, although it is sensitive to the data and works considerably better on simple tables. This extraction approach will support high quality retrieval for simple text tables, but some of the complex tables found in government data remain a challenge.

Future work will concentrate on improving the recall of table header lines and improving the models for answer retrieval. It is likely that our training set is still too small and not diverse enough to produce a table extractor for all types of data. Including WSJ documents along with the FedStats documents would be a good starting point for expanding the training data. Another interesting method worth trying would be a re-extraction with emphasis on headers from the documents that are more likely to contain the answer. Also, information in document may be exploited to improve header identification.

Smoothing could be a source of improvement for the retrieval models. One method would be to incorporate information from nearby paragraphs into the language model for the cell documents. Another proposal would build many different models for the cell. In addition to the cell documents we now create, various models for the text around the cell would also be generated, e.g. the text above the cell. In this way, metadata missed by the original extraction may be captured by the other models.

7. ACKNOWLEDGMENTS

This work was supported in part by the Center for Intelligent Information Retrieval, in part by the National Science Foundation under NSF grant #EIA-9983215. We appreciate the comments made by the reviewers of this paper, and colleagues from the CIIR. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

8. REFERENCES

Berger A and Lafferty J (1999) Information retrieval as statistical translation. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR ‘99). ACM, New York. pp. 222-229.

Byrd RH, Nocedal J and Schnabel RB (1994) Representations of quasi-Newton matrices and their use in limited memory methods. Mathematical Programming, 63:129–156.

Hurst M (2000) The interpretation of tables in texts. Ph.D. Thesis. University of Edinburgh, School of Cognitive Science, Informatics.

Hurst M (2001) Layout and language: an efficient algorithm for text block detection based on spatial and linguistic evidence. In: Proceedings of Document Recognition and Retrieval VIII. pp. 56–67.

Hurst M (2000) Layout and language: An efficient algorithm for text block detection based on spatial and linguistic evidence. In: Proceedings of the 18th International Conference on Computational Linguistics. July 2000.

Hurst M and Nasukawa T (1995) Layout and language: integrating spatial and linguistic knowledge for layout understanding tasks. In: Proceeding of the 18th International Conference on Computational Linguistics. pp. 334-338.

Krovetz R (1993) Viewing morphology as an inference process. In: Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, New York. pp. 191-202.

Lafferty J, McCallum A and Pereira F (2001) Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proceedings of the 18th International Conference on Machine Learning. pp. 282-289.

Malouf R (2002) A comparison of algorithms for maximum entropy parameter estimation. In: the Sixth Workshop on Computational Language Learning (CoNLL), 2002.

McCallum A (2002). Mallet: A Machine Learning for Language Toolkit, http://mallet.cs.umass.edu/

McCallum A, Freitag D and Pereira F (2000) Maximum entropy Markov models for information extraction and segmentation. In: Proceedings of the 17th International Conference on Machine Learning. pp. 591–598.

Ng HT, Kim CY and Koo JLT (1999) Learning to recognize tables in free text. In: Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics. pp. 443–450.

Ogilvie P and Callan J (2002) Experiments using the Lemur toolkit. In: Proceedings of the 2001 Text Retrieval Conference (TREC 2001). pp. 103-108. http://www-2.cs.cmu.edu/~lemur/

Pinto D, Branstein M, Coleman R, King M, Li W, Wei X, and Croft WB (2002) QuASM: A system for question answering using semi-structured data. In: Proceedings of the JCDL 2002 Joint Conference on Digital Libraries. pp. 46-55.

Ponte J and Croft WB (1998) A language-modeling approach to information retrieval. In**:** Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, New York. pp. 275-281.

Pyreddy P and Croft WB (1997) Tintin: A system for retrieval in text tables. In: Proceedings of the 2nd International Conference on Digital Libraries. pp. 193-200.

Rabiner LR (1990) A tutorial on hidden Markov models and selected applications in speech recognition. In: A. Weibel and K.-F. Lee, eds. Readings in Speech Recognition. Morgan Kaufmann, 1990. pp. 267–296.

Sha F and Pereira F (2003) Shallow parsing with conditional random fields. In: Proceedings of Human Language Technology, NAACL. pp. 213-220.

Voorhees E (2000) Overview of the TREC-9 question answering track. In: Proceedings of the 9th Text Retrieval Conference (TREC-9), 2000. pp. 71-80.

Zhai C and Lafferty J (2001) A study of smoothing methods for language models applied to ad hoc information retrieval. In: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, New York. pp. 334-342.

9. APPENDIX

A. Example questions for the FedStats data set:

1. What were seasonally adjusted durable goods sales in 1996?
2. What was the area planted with honeydews in 1997?
3. What was the value, in thousands of dollars, of the eggplant crop in 1998?
4. What percentage of children in Montana was at or below 200% of the poverty line between 1996 and 1998?
5. What was the percentage of fat in milk produced in 1993?
6. How many hours did New Hampshire power plants operate in the first quarter of 2001?
7. What is the phone number of Bering Air?
8. How many million tons of residential glass waste were generated in 1994?
9. What were total receipts for Japanese owned firms in 1992?
10. What percentage of minority firms were owned by blacks in 1992?
11. How many genes are on the map of chromosome 3?
12. What was the 1998 budget for conservation programs?
13. What was the value in millions of dollars of shipments of harvesting machines in 1989?
14. How many millions of board feet of hardwood lumber were produced in Maine in 1995?
15. What is the product code for Lighting and electronic glassware?

B. Example questions for the WSJ data set:

1. What was the GNP in the fourth quarter of 1986?
2. What was the value of the West German mark (in dollars) on 3/4/87?
3. What is the average cost of room and board at a private university?
4. What percent of Textron's business came from Aerospace technology in 1984?
5. What is the annual cost of technical support for a PC?
6. What was the US trade deficit with western Europe in June '88?
7. What was the new jobless claims total for the week ended March 19?
8. How many thousands of barrels of crude oil did Nigeria produce per day in 1987?
9. What was the date Lotus Modern Jazz became available?
10. How many Isuzu Troopers were imported as trucks?