Table Extraction Using Conditional Random Fields

David Pinto, Andrew McCallum, Xing Wei, W. Bruce Croft
Center for Intelligent Information Retrieval
University of Massachusetts Amherst
140 Governors Drive
Amherst, MA 01002
{pinto,mccallum,xwei,croft}@cs.umass.edu

ABSTRACT

The ability to find tables and extract information from them is a necessary component of data mining, question answering, and other 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 present difficulties for traditional language modeling techniques, however. This paper presents the use of conditional random fields (CRFs) for table extraction, and compares them with hidden Markov models (HMMs). Unlike HMMs, CRFs support the use of many rich and overlapping layout and language features, and as a result, they perform significantly better. We show experimental results on plain-text government statistical reports in which tables are located with 92% F1, and their constituent lines are classified into 12 table-related categories with 94% accuracy. We also discuss future work on undirected graphical models for segmenting columns, finding cells, and classifying them as data cells or label cells.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Experimentation, Theory, Measurement

Keywords

Tables, conditional random fields, hidden Markov models, information extraction, metadata, question answering.

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 ultimate example of conveying meaning through layout is the table.

Tables—textual tokens laid out in tabular form—are often used to compactly communicate information in fields and records. They have been described as "databases designed for human eyes." Tables appear in the earliest writing on clay tablets, and in the most modern Web pages. Some make use of line-art, others rely on whitespace only. They sometimes consists 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.

In previous work building the QuASM system (Question Answering Using Semi-structured Metadata)[13], extracting table data proved to be the most challenging task faced. Question answering (QA) systems typically perform a two-step retrieval in finding the information relevant to a query. First, documents are retrieved that match the question, then those documents are searched for possible answers. QuASM used a heuristic program to turn each data cell and its metadata into its own small document. That small document could be more accurately retrieved and quickly searched. It was found that the amount and quality of the metadata was extremely important to the success of the system, especially in the document retrieval step. Unfortunately, QuASM tended to extract too much information from the tables, which hurt performance.

To improve the extraction, and by extension the ability to answer questions, a move away from heuristics and toward a more formal model is proposed. Although there has been significant effort and progress in statistical language modeling for information retrieval tasks, there has been relatively little work in models that combine language and layout. Traditional language modeling, focused on a stream of language content, simply does not apply to data that is a two-dimensional mixture of content and layout features.

This paper presents a model of table extraction that richly integrates evidence from both content and layout by using *conditional random fields* (CRFs) [7]. CRFs are discriminatively-trained undirected graphical models that have great freedom to use complex, overlapping and non-independent feature sets that operate at multiple levels of granularity and multiple modalities. We describe a method that simultaneously locates tables in plain-text government statisti-

SIGIR'03, July 28–August 1, 2003, Toronto, Canada.

cal reports, and labels each of their constituent lines with tags such as header, sub-header, data, separator, etc. The features measure aspects of the input stream such as the percentage of alphabetic characters, the presence of regular expression matching months or years, and the degree to which whitespace in the current line aligns with whitespace in the previous line. In experiments on government reports, tables are located with 92% F1, and lines are labeled with 94% accuracy—reducing error by 80% over a similarly-configured hidden Markov model with the same features. For application to our question-answering system, we currently use these tags in concert with subsequent heuristics to find table cell boundaries, classify cells as data or label, and associate data cells with their corresponding label cells.

This paper also describes ongoing work on a more complex conditional random field that operates both vertically and horizontally, uses approximate inference procedures [17], and provides a complete conditional probabilistic model for table detection, cell segmentation and classification.

2. TABLE EXTRACTION

2.1 Related Work

Matthew Hurst [3, 6, 5, 4] 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, then determines what the blocks represent—using generative language models in both stages.

An example of a purely layout-based approach is found in Pyreddy and Croft [14]. A character alignment graph (CAG) is used to find text tables in documents in order to extract those tables for indexing for information retrieval. The CAG abstracts the table to text characters and spaces and attempted to identify titles, captions and data rows base on the structures revealed by the CAG. Since the whole table is extracted, there is no attempt to identify the fine elements of the table, only the sections of the table (header area, data area). Ng, Lim and Koo [12] apply machine learning techniques to identify rows and columns in tables, but again were only interested in finding the location of the table in the text not extracting its different components, either in terms of lines or cells.

Pinto et al. [13] build on the CAG with a heuristic method to extract the individual cells for QA. As this system extracts cell data and associates it with its 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 system does not finely distinguish between types of header rows, it tended to draw in extraneous metadata. Study of the errors made by the QA system show that the extraneous metadata is a leading cause of failure to retrieve the appropriate documents.

2.2 Task and Approach

Table extraction may be broken down into six overlapping sub-problems:

1. Locate the table.
2. Identify the row positions and types.
3. Identify the column positions and types.
4. Segment the table into cells.
5. Tag the cells as data or headers.
6. Associate data cells with their corresponding headers.

This paper concentrates on items one and two, employing a conditional probability Markov model for labeling the lines of a document. The labels indicate whether or not the line is part of a table, and if so, what role it plays in the table. The use of finite-state Markov models helps represent useful context, for example the fact that titles are expected to be followed by table headers, followed by data rows, followed by footnotes. State transition probabilities are a function of various features derived from the text.

Web pages contain two types of tables. Text tables are human formatted, generally using white space and fixed fonts to align columns. HTML tables are machine formatted, using a markup language to designate the position of cells. Even though not all HTML table elements are actually used for tables, text table extraction is the harder problem, since the markup has to be inferred from the layout of the table. Our source for these tables is a crawl of www.FedStats.com from July of 2001. These documents are rich in numeric data tables on a variety of subjects. A set of these documents was randomly selected and hand labeled for the experiments discussed in this paper.

An example of a text table is shown in Figure 1. This table demonstrates many of the challenges that tables present. Examine the sixth data line, labeled Sprouts. To correctly extract the 3rd number on the line (3200), we need to associate it with the three title lines, the super-header "Area Planted", the column header "1999", the sub-header "acres", the section-header "Brussels" and the row-header "Sprouts". Close examination shows that "Area Planted" is almost entirely over the second data column. Why not associate it only with that column? Why would the word "Brussels" be part of the sprouts data row instead of an empty data row? In Pinto et al., this problem was handled by including metadata generously. In addition to extracting the row and column headers, any row labeled as a header was included in its entirety. One of the goals of this work is to reduce that extraneous metadata to improve retrieval on a QA task.

Note that data extraction from HTML tables is not trivial, but presents a different set of problems. The table, rows and cells are delineated by the HTML markup, so finding a table is not difficult. However, HTML tables are often used to format documents instead of presenting data—proving once again that the combination of language and layout is important in extracting information from tables. A content judgment must be made to see if the table is worthy of having its data processed. The use of mark up is not consistent, 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 tables.

3. CONDITIONAL RANDOM FIELDS

Conditional Random Fields [7] 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*,

Figure 1: Example Table Snippet

```
<TITLE>               Principal Vegetables for Fresh Market:  Area Planted and Harvested
<TITLE>                                by Crop, United States, 1997-99 1/
<TITLE>                                          (Domestic Units)
<SEPARATOR>     ------------------------------------------------------------------------------
<SUPERHEADER>                   :          Area Planted          :        Area Harvested
<SUPERHEADER>         Crop      :--------------------------------------------------------------
<TABLEHEADER>                   :   1997   :   1998   :   1999   :   1997   :   1998   :  1999
<SEPARATOR>     ------------------------------------------------------------------------------
<SUBHEADER>                     :                              Acres
<SEPARATOR>                     :
<DATAROW>       Artichokes 2/   :    9,300      9,700      9,800      9,300      9,700      9,800
<DATAROW>       Asparagus 2/    :   79,530     77,730     79,590     74,030     74,430     75,890
<DATAROW>       Beans, Lima     :    2,700      3,000      3,200      2,500      2,000      2,900
<DATAROW>       Beans, Snap     :   90,260     94,700     98,700     82,660     87,800     90,600
<DATAROW>       Broccoli 2/     :  130,800    134,300    137,400    130,800    134,300    137,300
<SECTIONHEADER> Brussels        :
<SECTIONDATAROW> Sprouts 2/     :    3,200      3,200      3,200      3,200      3,200      3,200
<DATAROW>       Cabbage         :   77,950     79,680     79,570     75,230     76,280     74,850
```

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) [10] that avoid the *label-bias problem* [7].

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

$$p_\Lambda(\mathbf{s}|\mathbf{o}) = \frac{1}{Z_\mathbf{o}} \exp\left(\sum_{t=1}^{T}\sum_{k} \lambda_k f_k(s_{t-1}, s_t, \mathbf{o}, t)\right),$$

where $Z_\mathbf{o}$ is a normalization factor over all state sequences, $f_k(s_{t-1}, s_t, \mathbf{o}, t)$ 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 $\mathbf{o}$ 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 $-\infty$ to ∞.

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

$$p_\Lambda(\mathbf{l}|\mathbf{o}) = \sum_{\mathbf{s}:l(\mathbf{s})=\mathbf{l}} p_\Lambda(\mathbf{s}|\mathbf{o}),$$

where $l(\mathbf{s})$ is the sequence of labels corresponding to the labels of the states in sequence $\mathbf{s}$.

Note that the normalization factor, $Z_\mathbf{o}$, (also known in statistical physics as the *partition function*) is the sum of the "scores" of all possible state sequences,

$$Z_\mathbf{o} = \sum_{\mathbf{s}\in\mathcal{S}^T} \exp\left(\sum_{t=1}^{T}\sum_{k} \lambda_k f_k(s_{t-1}, s_t, \mathbf{o}, t)\right),$$

and that 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. The details are given in the following subsection.

3.1 Efficient Inference in CRFs

As in *forward-backward* for hidden Markov models (HMMs) [15], the probability that a particular transition was taken between two CRF states at a particular position in the input sequence can be calculated efficiently by dynamic programming. We can define slightly modified "forward values", $\alpha_t(s_i)$, to be the probability of arriving in state s_i *given* the observations $\langle o_1, ...o_t \rangle$. We set $\alpha_0(s)$ equal to the probability of starting in each state s, and recurse:

$$\alpha_{t+1}(s) = \sum_{s'} \alpha_t(s') \exp\left(\sum_{k} \lambda_k f_k(s', s, \mathbf{o}, t)\right).$$

The backward procedure and the remaining details of Baum-Welch are defined similarly. $Z_\mathbf{o}$ is then $\sum_s \alpha_T(s)$. The Viterbi algorithm for finding the most likely state sequence given the observation sequence can be correspondingly modified from its original HMM form.

3.2 Training CRFs

The $\Lambda = \{\lambda...\}$ weights of a CRF are typically set to maximize conditional log-likelihood, L, of labeled sequences in some training set, $\mathcal{D} = \{\langle \mathbf{o}, \mathbf{l} \rangle^{(1)}, ...\langle \mathbf{o}, \mathbf{l} \rangle^{(j)}, ...\langle \mathbf{o}, \mathbf{l} \rangle^{(N)}\}$:

$$L = \sum_{j=1}^{N} \log\left(p_\Lambda(\mathbf{l}^{(j)}|\mathbf{o}^{(j)})\right) - \sum_k \frac{\lambda_k^2}{2\sigma^2}$$

where the second sum is a Gaussian prior over parameters, with variance σ^2, that provides smoothing to help cope with sparsity in the training data [2].

When the training labels make the state sequence unambiguous (as they often do in practice), the likelihood function in exponential models such as CRFs is convex, so there are no local maxima, and thus finding the global optimum is guaranteed. It is not, however, straightforward to find it quickly. Parameter estimation in CRFs requires an iterative procedure, and some methods require fewer iterations than others.

Although the original presentation of CRFs [7] described training procedures based on iterative scaling [7], it is significantly faster to train CRFs and other "maximum entropy"-style exponential models by a quasi-Newton method, such as L-BFGS [1, 8, 16]. This method approximates the second-derivative of the likelihood by keeping a running, finite window of previous first-derivatives. Sha and Pereira [16] show that training CRFs by L-BFGS is several orders of magnitude faster than iterative scaling, and also significantly faster than conjugate gradient.

L-BFGS can simply be treated as a black-box optimization procedure, requiring only that one provide the first-derivative of the function to be optimized. Assuming that the training labels on instance j make its state path unambiguous, let $\mathbf{s}^{(j)}$ denote that path, then the first-derivative of the log-likelihood is

$$\frac{\delta L}{\delta \lambda_k} = \left(\sum_{j=1}^{N} C_k(\mathbf{s}^{(j)}, \mathbf{o}^{(j)}) \right) - \left(\sum_{j=1}^{N} \sum_{\mathbf{s}} p_\Lambda(\mathbf{s}|\mathbf{o}^{(j)}) C_k(\mathbf{s}, \mathbf{o}^{(j)}) \right) - \frac{\lambda_k}{\sigma^2}$$

where $C_k(\mathbf{s}, \mathbf{o})$ is the "count" for feature k given $\mathbf{s}$ and $\mathbf{o}$, equal to the sum of $f_k(s_{t-1}, s_t, \mathbf{o}, t)$ values for all positions, t, in the sequence $\mathbf{s}$. The last term is the derivative of the Gaussian prior.[1]

The upper parenthesized term corresponds to the expected count of feature k *given that the training labels are used to determine the correct state paths.* The lower parenthesized term corresponds to the expected count of feature k *using the current CRF parameters,* Λ*, to determine the likely state paths.* Matching simple intuition, notice that when the state paths chosen by the CRF parameters match the state paths from the labeled data, the derivative will be zero.

When the training labels do not disambiguate a single state path, expectation-maximization can be used to fill in the "missing" state paths.

4. TABLE EXTRACTION WITH CRFS

4.1 Line Labels

Our approach to table extraction starts by labeling each line of a document with a tag that describes that line's function relative to tables. The set of labels we use was designed by examining a large number of tables in Web documents. A good labeling accomplishes two goals; it marks the boundaries of the tables (table location) and identifies the row types useful for question answering and other applications. This section explains each label.

[1]We also tried an L1-like hyperbolic prior, $\sum_k (a/b)\log(\cosh(b\lambda_k))$, with slope a and sharpness b, and having derivative $ab\tanh(\lambda_k)$, but found it not to make a significant difference in accuracy.

4.1.1 Non-extraction Labels

Non-extraction labels represent lines that do not contribute information about table cells. The three labels 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 may appear anywhere in a document.

4.1.2 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. The header labels are TITLE, SUPERHEADER, TABLEHEADER, SUBHEADER and SECTIONHEADER. TITLE represents lines of text in which all content should be associated with every data cell in the following 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 cells 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.

4.1.3 Data Row Labels

Data row labels mark rows that contain data cells, the atomic information to extract. Data rows often contain header information for the row, also. The data row labels are 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.

4.1.4 Caption Labels

Caption labels mark rows that appear below data but still apply to the table. The caption labels are 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.

4.2 Feature Set

Pinto et al. [13] describe features used by a heuristic table extractor to identify a narrower range of line types. This work serves as a starting point for the features used in this paper. One difference, however, is the treatment of separator characters, such as dashes. During the creation of a CAG, these characters are treated as white space. The belief that separator characters convey a different meaning than white space prompted their inclusion in these experiments. As the system was developed other features were added to improve performance on development data.

4.2.1 White Space Features

White space is employed in documents and tables to improve readability. Common uses are to separate table cells, indent titles, indent sub-section data rows and to provide

a separation between lines of text. For purposes of this research white space is any character matching the regular expression "\s" as defined in the Java pattern class. The white space features:

- At least *four consecutive white space characters* are found in data rows, separating row headers from data, and in titles that are centered.
- At least *four space indents* are found in title lines, and often in header lines where the row header column is not labeled, or at the start of sub-headers and super-headers.
- Gaps, at least two consecutive white spaces between non-space characters, are often used to separate cells in data and header lines. At least *two gaps* is used as an indicator of a table line.
- A *large gap* of at least five consecutive white spaces is sometimes used in tables with a small number of columns to separate the row header from the data.
- A *single space indent* is often found in section data rows, as the indent sets these rows off from normal data rows.
- *All space characters* is a feature of a line that would match the regular expression ^\s*$, a blank line.
- The *percentage of white space* from the first non-white space character on can separate data rows from prose. Data rows tend to have a higher percentage of white space.

4.2.2 Text Features

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

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

4.2.3 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 look directly at punctuation:

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

4.2.4 Feature Representation

Each feature can be represented as a binary value. A 1 indicates the presence of the feature and a 0 indicates the lack of a feature. For percentage features, a threshold is set, above which the feature will score a one. Thresholds are listed in Table 1.

Feature	Threshold
Percentage of White Space	0.3
Header Features	0.6
Alphabet Characters	0.6
Digit Characters	0.7
Separator Characters	0.8

Table 1: Thresholds for Percentage Features

With CRFs, it is as easy to use continuous-valued features as it is to use discrete-valued ones. This enabled experiments in which the percentage features are not treated as binary features, but in which the actual percentages are the input.

4.2.5 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 value of features on one line are multiplied by the value of features on another (or the same) line, creating a new feature (Feature1&Feature2). Conjunctions help capture relationships that a linear combination of features may not. In these tests, the conjunctions used were the current line with the previous line, the current line with the following line and the conjunction of the two following lines together.

4.3 Data Set

We gathered a large set of documents from a crawl of www.FedStats.gov performed in June 2001. Documents were chosen for these experiments based on an earlier heuristic algorithm indicating that these documents may contain tables. From this set, documents were chosen randomly. This set contains documents both with and without tables.

Each line of the documents was labeled with one of the twelve labels described in section 4.1. A simple heuristic program was run to make an educated guess for a label for each line. Human judgment was then employed to correct mistakes in the heuristic labeling.

The first fifty-two documents labeled became the training set. These fifty-two documents contain 31,915 lines of text and 5,764 table lines. The next six documents labeled were set aside as a development set. These six documents contain 5,817 lines of text and 3,607 table lines. Finally, 62 more documents were labeled, containing 26,947 lines of text and 5,916 table lines. These documents were held out as final test data.

5. EXPERIMENTAL RESULTS

5.1 Alternative Models

We compare CRFs with two alternative models, both of which are configured to use the same (binary) feature set described previously. CRFs combine the benefits of finite state models and conditionally-trained log-linear models. The alternatives models are chosen to demonstrate importance of integrating both benefits.

Hidden Markov models [15], like CRFs, are Markov models; however they are trained to maximize the generative, joint probability of observation sequence and the label (state) sequence, rather than the conditional probability of the label sequence given observation sequence. We capture the multiple binary observation features with a smoothed multivariate Bernoulli [11].

The second alternative, a Maximum Entropy classifier, is a log-linear model trained to maximize a conditional probability—like the CRF. However it does not represent any finite-state context; it instead classifies each line independently. As with the CRF, we use a Gaussian prior, and train using L-BFGS.

5.2 Training CRFs

Using our Java implementation [9], we trained two versions of the CRF by L-BFGS. One version used only binary features (CRF Binary), and converged in 147 iterations. Another version used the actual value for the percentage features (CRF Continuous), but all other features were treated as binary, converging in 188 iterations. Both versions used a Gaussian prior and the same conjunction of features (see section 3.2)

5.3 Evaluation

5.3.1 Table Line Location

The accuracy of locating tables is evaluated by the F-measure, $(2 \times Recall \times Precision)/(Recall + Precision)$. [12] In this case, recall is the number of lines correctly labeled as belonging in a table (all but NONTABLE, BLANKLINE and SEPARATOR) divided by the actual number of table lines. Precision uses same numerator with the total number of lines labeled as table lines by the program as the denominator. Table 2 shows the results for the HMM and two CRFs. CRF Binary used all binary features while CRF Continuous represented the percentage features as the actual percentage.

Data Set	HMM	Max Ent	CRF Binary	CRF Continuous
Training	.813	.991	.998	.999
Development	.955	.981	.994	.993
Test	.648	.887	.912	.918

Table 2: Table Line Location, F-Measure

5.3.2 Line Identification

During development of the system, a small evaluation set was used to see if performance was improving. A larger set of data was held out for final testing, and no evaluations were done on that set until the best trained CRF was decided upon. The results for the development and test data are presented in Table 3. All four processes do well on the development set, but the superiority of the CRF shines through on the test data. In looking at the test data, a number of the documents had an unusual construction, where there were many blank lines between lines of text. The HMM mistakenly recognized these NONTABLE lines as TITLES. The CRF handled these more gracefully.

It is also useful to examine where the CRF made errors. Table 4 presents recall and precision for each of the labels for the best performing CRF, CRF Continuous, run on the test data. In this data set, labeling of TABLEHEADER lines was especially poor and especially troubling because the majority of mislabeled TABLEHEADER lines were as NONTABLE or DATAROW. This contrasted from the development set, where TABLEHEADERS were mostly missed as other header lines. SECTIONDATAROW lines were most often missed as DATAROW lines, although the DATAROW lines were most often missed when the table was missed, so NONTABLE is the most frequent mistake. A number of NONTABLE lines were mislabeled as TABLECAPTION, and these mislabels did not occur in the context of the end of a table. This aspect of the CRF needs to be more fully explored.

Data Set	HMM	Max Ent	CRF Binary	CRF Continuous
Training	89.7	99.5	99.9	99.9
Development	85.5	94.5	96.0	95.9
Test	65.4	85.4	93.4	93.5

Table 3: Percent of Lines Labeled Correctly

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

Table 4: CRF Continuous on Test Set

Figure 2 shows two examples of table labeling gone bad. The first is an example of a table of contents. Notice that there is no gap structure. Nor is there a series of lines with the same structure to indicate a body of a table. The *four consecutive periods* feature alone did not prove strong enough to label the data rows. With no data rows, there was no table. This may be an example of over fitting the training data. While there are two examples of this type of table in the training set, the sample is small compared to other type of tables. Additionally, the *four consecutive periods* feature most often occurs in the training set with data rows containing gaps.

The second table is partially correct. However, a number of rows lack gaps (two spaces) between cells, and those lines are labeled TABLEHEADER. Because there is a discernible table structure, however, the CRF does a better job of assigning a table-type label to the line. It's clear from looking at this data, gaps are over weighted in determining data rows. An examination of the training data shows only two lines labeled as a data row where digit cells are only separated by one space.

The results were also compared with the heuristic results based on the methods of Pinto, et al. That system uses

Figure 2: Problem Tables

```
Viterbi Label
NONTABLE                                    Index
NONTABLE                                                        Page
BLANKLINE
NONTABLE       Marketings, Income, and Value of Milk Production:
NONTABLE       United States, 1997-99 . . . . . . . . . . . . . . . . . . . . . . 3
BLANKLINE
NONTABLE       Milk and Cream:  Marketings and Income,
NONTABLE         by State and United States, 1998 . . . . . . . . . . . . . . . . 6
BLANKLINE
NONTABLE       Milk and Cream: Marketings and Income,
NONTABLE         by State and United States, 1999 . . . . . . . . . . . . . . . .10

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

TABLEHEADER                        Item                                          All   H    B    O    W   NMM
BLANKLINE
SEPARATOR      ------------------------------------------------------------  ---- ---- ---- ---- ---- ----
BLANKLINE
NONTABLE       In '96, Owns the Same Business Owned in 1992@4: Yes           68.9 67.3 60.1 71.2 66.0 70.5
BLANKLINE
TABLEHEADER    In '96, Owns the Same Business Owned in 1992@4: No            25.5 26.9 32.1 23.2 27.7 24.2
BLANKLINE
DATAROW        In '96, Owns the Same Business Owned in 1992@4: Not Rptd       5.7  5.7  7.7  5.6  6.3  5.3
BLANKLINE
DATAROW        Year That Ownership Ended@4: 1994                              5.4  4.9  6.3  5.1  5.3  5.5
BLANKLINE
DATAROW        Year That Ownership Ended@4: 1995                              4.5  4.9  5.3  4.7  4.7  4.4
BLANKLINE
DATAROW        Year That Ownership Ended@4: Not Reported                      9.9 10.1 12.0  9.0 11.0  9.3
BLANKLINE
TABLEHEADER    Year That Ownership Ended@4: N/A@5                            68.9 67.3 60.1 71.2 66.0 70.5
```

four names (captions, headers, data and non-table) to mark lines, but our labels map nicely to those. Based on the more generous measure given by the smaller number of labels the CRF outperformed the heuristic program (see Table 5). This is encouraging, since improvements on a heuristic system requires constant changes to program code, while improvements on the CRF system require only defining new features that can be automatically extracted. These results also show that among the three methods discussed, CRFs have the most consistent performance.

Data Set	Heuristic	CRF Continuous
Development	77.1	98.6
Test	92.0	95.3

Table 5: Percent of Lines Labeled Correctly, Heuristic Measurement

6. FUTURE WORK IN TABLE INFORMATION EXTRACTION

The conditional random field model described above locates tables and tags their constituent lines, but knows nothing of columns, does not segment individual table cells horizontally or vertically, and does not tag individual cells as being data or header. In ongoing work we are implementing a richer conditional random field that will solve all these tasks by using a graphical model with nodes at multiple levels of granularity, and with dependency-edges running both vertically and horizontally.

In this new model there are nodes representing the tag of each line of text (as in the model described above), horizontally-connected nodes for each individual character[2] (with labels indicating horizontal segmentation of cells), and vertically-connected nodes for each individual character (with labels indicating vertical segmentation of cells). The latter two types of nodes use standard `OIB`-style segmentation labels to indicate the `Begin`, `Interior`, and `Outside`, of cells—with different families of these tags to indicate data cells versus header cells.

Thus, for example, a character node might be tagged with `Vertical-Begin/Data` and `Horizontal-Interior/Data` to indicate the it is at the top of a data cell vertically, and in the middle of the data cell horizontally.

This model is connected in a grid-pattern instead of a linear-chain, and the dynamic-programming-based efficient inference described above is no longer possible. Gibbs sampling or loopy belief propagation would both apply, but we rely instead on a procedure for exact MAP estimates based on tree agreement [17]. Inference in our model consists of three steps performed in a cycle. First, as described above, belief propagation is performed on a linear-chain-shaped CRF (with one node per line), in order to tag each line of text. Second, belief propagation is performed using the horizontal edges of the character nodes. Third belief propagation is performed using the vertical edges of the character nodes. The overall inference procedure then cycles to step 1, where the tagging decisions of previous inference steps can now be examined as features. The procedure is guaranteed to converge.

[2]or alternatively, each space-separated token

For efficiency, the character-level inference might only occur in regions of the document that the first step indicated were likely to contain a table. Association of data cells with their corresponding header cells may not require sophisticated modeling—for each data cell, simply finding header cells that intersect horizontally and vertically is expected to perform extremely robustly. Experimentation with this new model is pending.

While this model is being developed, experiments will also proceed on using the CRF labeling with the QuASM extraction program. Our hypothesis is that the finer labeling coupled with the column finding heuristics of the old program will solve the problem of extraneous metadata. We then hope to show that these cleaner documents perform better on a QA retrieval task.

7. CONCLUSIONS

Tables occur in many types of documents, in many formats and variations. They contain data that are important for various information retrieval tasks, including document retrieval, question answering, summarization and novelty detection. Processing of tables is distinguished from many other language processing tasks in that it requires modeling a complex, interdependent mixture of language and formatting features. For this reason, a conditionally-trained probability model is particularly well-suited to table extraction, because conditional models handle well many arbitrary, non-independent features of its input. CRFs offer the combined benefits of conditional-probability training and Markov finite-state context, and these benefits are shown in our experimental results to provide a practical improvement over generative models (an HMM) and conditionally-trained stateless models (a maximum entropy classifier).

There are many promising avenues for further work. Our current training set is not large. Since the models are clearly overfitting, simply training with more data may help significantly. There are also opportunities for adding more complex features. The most significant missing component of our model is the ability to locate and label cells within the table; our plans for this problem are outlined above. These models are expected to also be applicable to many other non-tabular tasks involving mixtures of content, formatting and layout.

8. ACKNOWLEDGMENTS

This work was supported in part by the Center for Intelligent Information Retrieval, the Central Intelligence Agency and the National Science Foundation under NSF grant #EIA-9983215, the Advanced Research and Development Activity under contract number MDA904-01-C-0984 and in part by SPAWARSYSCEN-SD grant numbers N66001-99-1-8912 and N66001-02-1-8903. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

We would like to thank Aron Culotta for his code optimizations and Fangfang Feng, Andres Corrada-Emmanuel and Stephen Cronen-Townsend for their technical help.

9. REFERENCES

[1] R. H. Byrd, J. Nocedal, and R. B. Schnabel. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming*, 63:129–156, 1994.

[2] S. F. Chen and R. Rosenfeld. A Gaussian prior for smoothing maximum entropy models. Technical Report CMU-CS-99-108, CMU, 1999.

[3] M. Hurst. *The Interpretation of Tables in Texts*. PhD thesis, University of Edinburgh, School of Cognitive Science, Informatics, 2000.

[4] M. Hurst. Layout and language: An efficient algorithm for text block detection based on spatial and linguistic evidence. In *Proc. Document Recognition and Retrieval VIII*, pages 56–67, 2001.

[5] M. Hurst. 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. ICCL*, July 2000.

[6] M. Hurst and T. Nasukawa. Layout and language: Integrating spatial and linguistic knowledge for layout understanding tasks. In *Proceeding of the 18th International Conference on Computational Linguistics. (COLING 2000)*, 2000.

[7] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*, 2001.

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

[9] A. McCallum. Mallet: A machine learning for language toolkit. http://www.cs.umass.edu/∼mccallum/mallet, 2002.

[10] A. McCallum, D. Freitag, and F. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proc. ICML 2000*, pages 591–598, 2000.

[11] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on "Learning for Text Categorization"*, 1998.

[12] H. T. Ng, C. Y. Kim, 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, 1999.

[13] D. Pinto, W. Croft, M. Branstein, R. Coleman, M. King, W. Li, and X. Wei. Quasm: A system for question answering using semi-structured data. In *Proceedings of the JCDL 2002 Joint Conference on Digital Libraries*, pages 46–55, 2002.

[14] P. Pyreddy and W. Croft. Tintin: A system for retrieval in text tables. In *Proceedings of the Second International Conference on Digital Libraries*, pages 193–200, 1997.

[15] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In A. Weibel and K.-F. Lee, editors, *Readings in Speech Recognition*, pages 267–296. Morgan Kaufmann, 1990.

[16] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology, NAACL*, 2003.

[17] M. J. Wainwright, T. Jaakkola, and A. S. Willsky. Exact MAP estimates by (hyper)tree agreement. In *Advances in Neural Information Processing (NIPS)*, 2003.