Lessons Learned in Automatically Detecting Lists in OCRed Historical Documents

Thomas L. Packer, David W. Embley
Department of Computer Science
Brigham Young University
Provo, Utah, USA
tpacker@byu.net

ABSTRACT

Lists are often the most data-rich parts of a document collection, but are usually not set apart explicitly from the rest of the text, especially in a corpus of historical OCRed documents. There are many kinds of lists, differing from each other in both layout and content. Writing individualized code to process all possible types of lists is an expensive challenge. In the present research, we focus on general list detection, the first step in a larger process of general list reading. Our system, ListDetector, automatically locates lists from noisy word labels obtained from cheaply developed dictionaries and regular expressions. In this paper, we start by describing a simple baseline system—the first system we are aware of to address general list detection in plain text or OCRed documents. From there, we present several of the challenges and corresponding solutions that we discovered as we raised the F-measure of ListDetector from 79% to 86.3%. We compute evaluation metrics against a gold standard corpus of OCRed documents in the family history domain that we have manually annotated for the tasks of list detection and structure recogniton. We will continue adding to this corpus and make it publically available for other researchers to use.

1. INTRODUCTION

Family history, among other kinds of research, depends on the discoverability of information recorded in unstructured and semi-structured text documents. Lists are often the most data-rich parts of a document collection. Before processing a list, e.g. extracting information from the text of the list, it is helpful to first automatically find and isolate that list from the surrounding text and infer its structure. However, lists are usually not set apart explicitly from the rest of the text, especially in a corpus of OCRed historical documents. Even if one kind of list is well delimited, there are often other kinds of lists—even in the same document—that are not, or at least not in the same way. Lists differ from each other in both layout and content. One list may be structured like a "bulleted list" of verbose statements delimited by hard returns. Another list may consist of a short sequence of names in a sentence delimited by commas. OCRed lists are especially inconsistent considering some documents contain little or no punctuation because of image noise removal.

We define list recognition as the process of finding and interpreting the structure of lists in text. List recognition provides structural cues to benefit downstream processes—processes that may target either the text inside or outside of the lists. It is analogous to the better-known process of table recognition. Hu et al. [7] decompose the task of table recognition into (1) table detection and (2) table structure recognition. We likewise decompose list recognition into (1) list detection and (2) list structure recognition. In this paper, we focus on general list detection in OCRed documents.

There are existing published efforts in processing specific types of lists in OCRed documents like bibliographies and tables of content. In such circumscribed applications, list detection is a small and straightforward addition to structure recognition. Writing individualized structure recognition code to process all possible types of lists would be much more challenging. As we target the "long tail" challenge in list detection, we prefer a general approach, one that is adaptive to the format, genre, and domain of each page encountered. A solution to general list detection in plain or OCRed text has never been published as far as we are aware. We now enumerate the most closely related research we have found.

Existing work shows how to recognize certain kinds of machine-printed lists using specialized knowledge, geometric table layout recognition techniques, or hand-crafted rules. Le and Thoma [9] and Green [5] use geometric table layout recognition approaches that we believe will not achieve high generality when targeting arbitrary lists without using additional textual and semantic clues. Like Green, Belaïd [1] recognizes tables of content which are similar to lists. His method relies on part-of-speech tagging and specialized rules which we believe make it less general and scalable. The "layout and language" perspective of Hurst and Nasukawa [8] is a general approach to logical document layout recognition using an n-gram language model (language) to regroup physical blocks of text identified using visual clues (layout). They indicate that this approach may help detect some lists. However, since most text in lists is not grammatical, we expect that it will be challenging to adapt Hurst's approach to arbitrary

To appear in the Family History Technology Workshop; February, 2012; Salt Lake City, Utah.

lists. The preceding work focuses on structure recognition which may contain an element of structure detection or categorization. We are not aware of work focusing on list detection in particular.

We also find work related to the present research in the web information extraction community, including the following. Dalvi et al. [3] detect groups of structured records in HTML documents based on patterns in the noisy field labels applied by inexpensive field recognizers. Groups of structured records could conceivably contain semi-structured lists as we propose to show in this paper, although it is not clear whether they target lists. Gupta and Sarawagi [6] apply a few heuristics to filter out navigational and verbose lists from a candidate pool consisting of all HTML lists in a web crawl. They rely on HTML list tags which help in detecting HTML lists. In their system, starting from all HTML lists in their corpus, they discard a list for any of the following reasons: (1) it has less than 4 records, (2) it has more than 300 records, (3) it has even one record more than 300 bytes long, (4) more than 20% of its records do not have delimiters, or (5) more than 70% of its records are contained inside anchor tags. Chang et al. [2] demonstrate an efficient algorithm for finding repeated substrings in an HTML page and use it to locate groups of structured records. Embley at al. [4] find and segment records in multi-record web pages using a combination of several heuristics looking for the best record delimiter. The web wrapper induction techniques above generally rely on repeated sequences of HTML tags which are not available in OCRed lists.

We are currently designing a system we call ListDetector to be an adaptive, weakly-supervised system for automatically detecting lists in OCRed documents, at a cost lower than typical supervised machine learning techniques allow. In the current implementation described here, ListDetector locates lists automatically after tuning hyper-parameters on a few pages of hand-annotated training data, relying greatly on weak supervision in the form of dictionary and regular expression matching. ListDetector starts processing a page by assigning labels to word tokens that match any of a set of cheaply constructed dictionaries and regular expressions. Given the resulting sequence of noisy labels in a page, ListDetector detects lists by identifying repeating patterns among subsets of labels.

This present work includes the following contributions. We provide the first version of a new public dataset containing images and corresponding OCR output and annotations for the pages in a variety of documents within the family history domain. We describe the data as it relates to the task of list detection in Section 2. We present ListDetector (Section 3), starting with a simple baseline version, followed by several of the challenges and corresponding solutions we encountered during development. We empirically evaluate ListDetector on the training data along the way. We present final experimental results in terms of a separate set of test pages in Section 4. This is the first proposal and evaluation of a general-purpose list detection algorithm we are aware of. We relate a number of additional lessons learned during the development of ListDetector in Section 5 and enumerate conclusions and future work in Section 6.

2. TASK AND DATA

We are currently collecting and annotating a corpus of several kinds of OCRed historical documents which we will provide to the public. For the current project, we manually marked the beginnings and endings of a variety of lists in a sample of pages from two historical newspapers and one high school yearbook. The newspapers are The South Australian Government Gazette, 1867–1884, and The Queensland Government Gazette, 1904–1910. The yearbook is the Bedford High School Abacus, Bedford, Ohio, 1960. We annotated three pages from each of the newspapers (which are all the pages we have available to us) and 36 pages from the yearbook. We then split these pages into a training set consisting of all six pages of newspaper and 20 of the pages of the yearbook, and a test set consisting of the remaining 16 pages of the yearbook. Many of the pages contain no lists, and all the pages contain at least some non-list text, so we have plenty of opportunity to test for false positives. We put all the newspaper pages into the training set because we had already begun development work while visually inspecting these pages before deciding to create a relatively "blind" test set . Figure 1 shows part of a yearbook page and Figure 2 shows the corresponding OCR text.

Before creating a system to detect lists, we should define what a list is. We are working on a formal definition to propose in future papers. For this paper, we enumerate the criteria that are most relevant to the discussions below.

A list is text consisting of three or more list items called records. Each record consists of one or more fields (substrings), in addition to the record delimiter if present. At least one field in each record must correspond in both semantic role or category and position to a field in some other record. At least one record must contain text that is different from the other records. These criteria must be visible in the OCR text to a person familiar with the corresponding image and the topic (domain) of the document.

Figure 3 shows our annotation of the two lists in Figure 2.

We experimentally evaluate several versions of ListDetector. When one approach predicts the location of a list, it effectively divides all the word tokens on a page into two categories: list and non-list. Because we compute accuracy metrics from word token counts, it is important to know how we tokenize the text. We split text into word tokens at whitespace boundaries and at transitions between character types. We consider alphabet, digit, and each type of punctuation character to be a separate character type.

We compute precision, recall, and F-measure using standard formulas borrowed from the information retrieval community. *Precision* is the number of words that ListDetector correctly assigns to a list divided by the total number of words it assigns to lists. *Recall* is the number of words ListDetector correctly assigns to a list divided by all words that should have been assigned to a list. *F-measure* (F_1) is the harmonic mean of precision (P) and recall (R):

$$F_1 = \frac{2PR}{P+R}$$

We compute these metrics for each page. We then compute aggregate F-measures over a set of pages in two ways: using macro-averaging and micro-averaging. A macro-average

Party committee — Judy Hyde, Jonna Wing, Don Sochacki, Peggy Buday and Janet Martin. They made the good time possible.

SENIOR CLASS PARTY

Clad in grass skirts, sarongs, shorts, jeans and bright shirts the seniors gathered for their annual party, aptly called "The Hawaiian I."

Figure 1: Part of a page from a school yearbook.

```
<line>Party committee—Judy Hyde Jonna Wing Don Soch
</line>
<line>ack Peggy Buday and Janet Martin They made the
</line>
<line>good time possible</line>
<line>SENIOR CLASS PARTY</line>
<line>Clad in grass skirts sTrongs shorts jeans and
bright</line>
<line>shirts the seniors gathered for their annual
party aptly</line>
<line>called The Hawaiian I</line>
```

Figure 2: OCR for image in Figure 1.

```
<text>Party committee—</text>
<list>
    <record>Judy Hyde</record>
    <record>Jonna Wing</record>
    <record>Don Soch ack</record>
    <record>Peggy Buday and</record>
    <record>Janet Martin</record>
</list>
<text>They made the good time possible SENIOR CLASS
PARTY Clad in<text>
<list>
    <record>grass skirts</record>
    <record>sTrongs</record>
    <record>shorts</record>
    <record>jeans and</record>
    <record>bright shirts</record>
</list>
<text>the seniors gathered for their annual party
aptly called The Hawaiian I</text>
```

Figure 3: Expected list detection and structure recognition output for the text in Figure 2.

over pages is simply the mean of the metric over those pages. A micro-average is a weighted average—weighted by the number of words in each page. Using both averages gives a more balanced impression of performance. The macro-average emphasizes the most common styles of page; the micro-average emphasizes the largest pages. To present a single score by which to compare approaches, we use the average of the macro- and micro-averaged F-measures, and call this the AA score.

3. APPROACHES TO LIST DETECTION

We designed our first approach to list detection with simplicity, our list definition, and the text in Figure 4 in mind. We call it *Simple Literal Pattern Area (SLPA)*. SLPA relies on the assumption that all the records in a list will have some amount of identical text in common. SLPA enumerates, scores, and ranks all the literal substrings ("patterns") in an input page. Based on our list definition, SLPA filters out any pattern candidate less than two words long or occurring less than three times. For example, the pattern "<NewLine> District No." has a length of four and a hit count of seven in Figure 4, so this pattern is preserved, along with all substrings longer than one word. SLPA scores the remaining patterns using the product of the pattern's length and hit count (the pattern's "area"). Our example pattern has an area score of 28. Finally, SLPA selects the pattern with the highest area score and marks all text between the first and last occurrence of this pattern as a list.

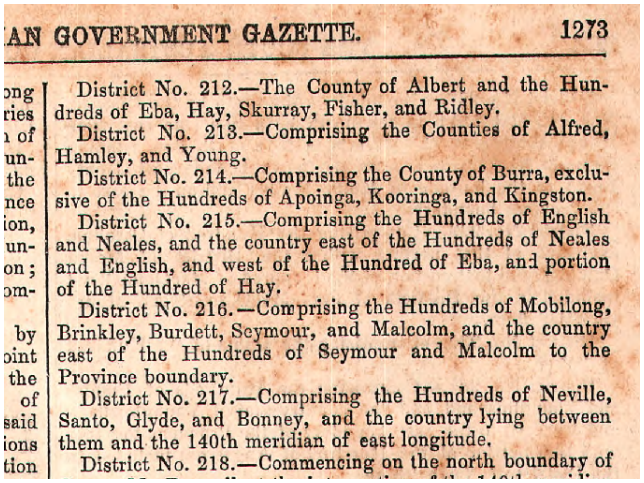
SLPA, as well as all variation reported in this paper, conveniently though incorrectly assume that the first occurrence of a pattern marks the beginning of a list and the last occurrence of a pattern marks the end of a list, which causes ListDetector to discard the last nine tokens in Figure 4. The approaches in this paper also recognize only contiguous and constant patterns. For example, if an OCR error had inserted or removed a period in one of the occurrences of "<NewLine> District No." in Figure 4, that occurrence would not be considered part of the same pattern. We will revisit these two limitations in future research.

The AA score for SLPA on the training data is 51.2%, as we show in the top row of Table 1. This score is our starting point, a baseline against which to compare the following developmental changes of ListDetector. One note of caution as we implement changes in response to errors we see in the training data: if an "improvement" is too specific to the training data, then scores computed on the test data may actually decrease at the same time they increase in the training data. This is called over-fitting the training data in the machine learning community.

SLPA relies on very lenient constraints on pattern length and hit count to filter spurious lists. As we reviewed the false positives produced by SLPA in the training data, we noticed that stricter constraints would help. Increasing the lower bound on pattern length and hit count independently improved precision, but usually at the expense of recall and F-measure. Raising the lower bound on their product from 6 to 10, however, improved both macro-averaged and micro-averaged F-measure which moves the AA score from 51.2% to 62.8% on the training data. We call this second approach *Bounded Literal Pattern Area (BLPA)*.

List Detection Approach	Macro-averages			Micro-averages			AA
	Prec.	Rec.	F1	Prec.	Rec.	F1	F1
Simple Literal Pattern Area (SLPA)	41.0%	57.7%	31.3%	62.3%	82.5%	71.0%	51.2%
Bounded Literal Pattern Area (BLPA)	77.7%	56.0%	49.6%	70.9%	82.1%	76.1%	62.8%
Repeated Literal Pattern Area (RLPA)	77.7%	61.8%	53.4%	73.1%	93.6%	82.1%	67.7%
Pattern Area, Naïve Bayes Selector (PA-NB)	**65.3%**	85.3%	62.7%	74.6%	96.2%	84.0%	73.4%
Pattern Area, Standard Deviation Selector (PA-SD)	42.7%	**98.7%**	46.6%	63.3%	**99.6%**	77.4%	62.0%
Bounded Pattern Area, Standard Deviation (BPA-SD)	55.7%	90.2%	57.3%	66.8%	98.4%	79.6%	68.5%
Bounded Pattern Area, Standard Deviation (BPA-SD) 2	63.6%	87.6%	62.0%	73.0%	93.8%	82.1%	72.0%
Bounded Pattern Area, Standard Deviation (BPA-SD) 3	64.5%	89.1%	**63.3%**	**76.9%**	96.9%	**85.7%**	**74.5%**

Table 1: Evaluation of incremental changes in ListDetector with respect to the *training data*. "AA" is the average of the two kinds of F-measure averages.



AN GOVERNMENT GAZETTE. 1273

District No. 212.—The County of Albert and the Hundreds of Eba, Hay, Skurray, Fisher, and Ridley.
District No. 213.—Comprising the Counties of Alfred, Hamley, and Young.
District No. 214.—Comprising the County of Burra, exclusive of the Hundreds of Apoinga, Kooringa, and Kingston.
District No. 215.—Comprising the Hundreds of English and Neales, and the country east of the Hundreds of Neales and English, and west of the Hundred of Eba, and portion of the Hundred of Hay.
District No. 216.—Comprising the Hundreds of Mobilong, Brinkley, Burdett, Seymour, and Malcolm, and the country east of the Hundreds of Seymour and Malcolm to the Province boundary.
District No. 217.—Comprising the Hundreds of Neville, Santo, Glyde, and Bonney, and the country lying between them and the 140th meridian of east longitude.
District No. 218.—Commencing on the north boundary of

Figure 4: Part of a page from a historical newspaper.

BLPA assumes that there is at most one list per page and that this list is spanned by just one pattern. As we inspected the training data, we saw that neither assumption is correct. Our next approach, *Repeated Literal Pattern Area (RLPA)*, addresses this deficiency by marking text spanned by the top three patterns instead of the top one pattern. It ignores patterns of intermediate rank that are covered by higher-ranking patterns. Using multiple patterns raises AA score in the training data from 62.8% to 67.7%.

RLPA and its predecessors have a glaring deficiency. According to our definition, a list's records must share one or more fields, but these fields need not contain identical text. Many of the lists in our training data contain no reasonable literal pattern. On the other hand, almost all of them contain semantically related fields that could at least supplement a literal pattern. Consider the three-digit numeral in the records of Figure 4 that join the four-word pattern mentioned above with an additional two- to three-word pattern. One way we might recognize a repeating pattern of semantically-related fields is to construct named entity recognizers for those fields. However, we wish to avoid this as it is expensive to develop accurate recognizers for the fields in each list in a corpus. Dalvi et al. [3] provide an idea that we adapt to the present setting. We construct cheap, noisy field recognizers in the form of flat dictionaries and regular expressions which have some chance at matching some of the fields in the lists of a given domain. ListDetector will add a label to all the words in a page that match a recognizer.

For the following experiments, we assembled 11 recognizers. Six dictionaries include 8400 given names, 142,000 surnames, 13 person titles, 200 Australian cities, eight Australian states, and 15 religions. Four regular expressions include numerals of between one and four digits in length, upper-case initial letters, with and without a following period, and capitalized words. It is important to note that aside from the category of these 11 recognizers, they were not refined in any way to perform well on this corpus—their contents were taken from external sources. We also include the word itself as a noisy label so that literal patterns are still discoverable. We now want ListDetector to find list-like patterns among this extended vocabulary of symbols just like it did with the vocabulary of the literal text.

Assembling enough field recognizers to ensure that every list in a corpus is sufficiently covered by recognizers means that often ListDetector will assign multiple labels to a given word. In our training data, it is common for the same word to receive four different labels (e.g. *given name*, *surname*, *capitalized word*, and the word itself) and have even seen words receive five labels (the four just mentioned plus *Australian city*). The primary challenge in using multiple, noisy labels is to construct a single, flat pattern consisting of just one label per token—the "correct" label—and to ensure that the resulting patterns are more indicative of lists than the original, literal text.

To address this challenge, we implemented two ways to select a single label per token. The first method is to train a naïve Bayes classifier to select each token's label based on the labels of that token and its six neighbors. We do not want to the incur the additional cost of hand labeling training data, so we train the classifier on the noisy labels, themselves, as both input features and output classes, training a separate model for each token using all the tokens in a page except for the token it is applied to. Our justification for using this approach is that, assuming there is a list on a page, the words in the list will tend to be in similar contexts (next to neighbors with similar labels) and therefore the label most strongly associated with that context should be chosen by the model. In other words, we expect that the true labels in

a list pattern will produce label co-occurrence counts that dominate any other labels that occur randomly. We call this approach *Pattern Area with Naïve Bayes Label Selector (PA-NB)*. It performs better than RLPA after tuning some hyper-parameters on the training set as described below for standard deviation approach, increasing the AA score from 67.7% to 73.4%.

Pattern Area with Standard Deviation Label Selector (PA-SD) replaces the naïve Bayes classifier with a simpler heuristic based on a similar justification. Assuming there is a list on a page containing at least one type of labeled field, the distance between subsequent instances of the label should tend to be constant, or in other words the standard deviation of the distances should be low. All other labels should tend to be placed randomly on the page and therefore have a higher standard deviation. This is inspired by a similar heuristic used by Embley et al. [4] to perform record boundary detection in web pages. To leverage this heuristic, ListDetector ranks the available field labels according to standard deviation and selects the best-ranked label for each token.

This second heuristic performs poorly at first, lowering the training data AA score from 67.9% to 62.0%. Using 8 patterns instead of 3 patterns per page improves the score only a small amount. Using the noisy labels has naturally improved recall, now that more list content can be used to generate patterns. We must therefore improve precision, which the noisy labels have decreased so much that the F-measure also decreases. Bounding the area score is one way to do this. Changing the lower bound from 10 to 26 increases the AA score to 68.5%. We call this approach *Bounded Pattern Area with Standard Deviation Label Selector (BPA-SD)*.

BPA-SD still produces a lower precision than recall. One reason is, two lists may share the same pattern and yet be separated from each other by some other non-list text which ListDetector classifies as being part of the same list. Another reason is, ListDetector can find patterns on a page without any list. It will select the ones with the lowest standard deviation no matter how high that standard deviation is. To address these two concerns and improve precision, we altered BPA-SD in two ways. In the first (BPA-SD 2), ListDetector removes text from a list between two instances of a pattern whenever they are farther apart than 2.5 times their pattern's standard deviation. In the second (BPA-SD 3), ListDetector will filter out patterns that have a standard deviation larger than 50 word tokens. These two final adjustments raise the AA score to 72.0% and 74.5%, respectively, the highest on the training data in this paper.

4. EXPERIMENTAL EVALUATION

Table 2 shows the results of intermediate and final versions of ListDetector as evaluated against the test data—pages that we did not inspect during development. The raising of the lower bound on pattern area in SLPA that improved precision in the training data without affecting recall did, in fact, affect recall substantially in the test data while at the same time maximizing precision. The next change did nothing to help the situation, namely increasing the number of selected patterns per page. The change did improve recall in the training data. We expect that when we have a larger test set, some improvement will again be visible. The test data also confirms that using either the unsupervised naïve Bayes classifier or the standard deviation heuristic to select noisy labels is effective at raising F-measure if used in conjunction with the additional precision-raising constraints.

5. ADDITIONAL LESSONS

Keeping with the theme implied by this paper's title, we would like to pass on a few other lessons learned (or relearned) as part of this project.

Extra blank lines and horizontal rules between text blocks appear like list patterns to ListDetector. We added them as unique "words" to our OCR text at first, wanting to preserve as much information about a page as we could. We removed them from the OCR before generating the evaluation metrics reported here.

Using a training set that is not representative of the test set can make it difficult or impossible to perform well (assuming no transfer learning). This is similar to changing the target evaluation metric in the middle of development. It is harder to hit a moving target than a stationary one.

It is sometimes unknown when positive changes in the training data will consistently predict positive changes in the test data. We were more confident that improvements in the training data were not accidental if there was an improvement in more than one metric, such as both macro-averaged and micro-averaged F-measure. We believed that using both metrics as a guide helped us accept enhancements in ListDetector that actually improved performance on the unseen test data.

In early experiments, we noticed that improving precision when the test set already had higher precision than recall was often a bad idea, even if the change improved the overall F-measure in the training data. Since F-measure stays close to the lower of its two components, this suggests that it is useful to keep track of which component is the limiting component at each point in development. This observation also reiterates the need to ensure that training and test data are representative of each other so it is less likely that precision is the limiting component in one set at the same time that recall is the limiting component in the other.

Also early in development, we noticed that jittery evaluation metrics (metrics that went up and down drastically in comparison to minor changes in hyper-parameter values) was a sign of a bug (a misplaced parenthesis) in one of our most important numerical functions (standard deviation).

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented the first proposed method of general list detection in free text or OCR output we are aware of. We have enumerated some of the challenges and possible solutions in this task and have empirically evaluated and compared these solutions. We conclude that, despite several deficiencies and naïve assumptions, ListDetector can categorize text as belonging to lists with 86.3% F-measure based on a micro-average over pages in our test corpus. We have also begun producing an annotated corpus of historical OCRed document for others to use.

Since we are just beginning to investigate list detection,

List Detection Approach	Macro-averages			Micro-averages			AA
	Prec.	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**	**F1**
Simple Literal Pattern Area (SLPA)	55.9%	43.8%	34.5%	74.9%	83.5%	79.0%	56.8%
Bounded Literal Pattern Area (BLPA)	**100.0%**	12.3%	12.4%	**100.0%**	63.2%	77.4%	44.9%
Repeated Literal Pattern Area (RLPA)	**100.0%**	12.3%	12.4%	**100.0%**	63.2%	77.4%	44.9%
Pattern Area, Naïve Bayes Selector (PA-NB)	79.4%	41.6%	38.3%	86.1%	85.2%	85.6%	62.0%
Pattern Area, Standard Deviation Selector (PA-SD)	34.5%	**81.7%**	36.0%	46.6%	**96.5%**	62.8%	49.4%
Bounded Pattern Area, Standard Deviation (BPA-SD)	63.0%	56.0%	42.4%	72.6%	84.6%	78.1%	60.2%
Bounded Pattern Area, Standard Deviation (BPA-SD) 2	83.7%	55.5%	**51.1%**	89.3%	83.4%	**86.3%**	**68.7%**
Bounded Pattern Area, Standard Deviation (BPA-SD) 3	83.7%	55.5%	**51.1%**	89.3%	83.4%	**86.3%**	**68.7%**

Table 2: Evaluation of incremental changes in ListDetector with respect to the *test data*. "AA" is the average of the two kinds of F-measure averages.

there are many additional improvements we plan to make to the research described above. Our experiments have been sensitive to the small size of both training and test data and to ListDetector's several hyper-parameters. We plan to overcome these challenges in our ongoing research. It is important to have a representative sample of pages in both training and test sets. We will therefore annotate hundreds of additional pages of OCRed text from a greater variety of historical documents. The annotations will contain information about list location, list structure, and relational field content.

We have also begun designing and implementing improvements and alternate approaches to list detection by integrating the following additional information and techniques into ListDetector:

1. Leveraging visual layout clues like line spacing, tab stops, and font style,
2. Acknowledging OCR errors by relaxing the strictness or contiguousness of our pattern language,
3. Using patterns discovered on one page to help discover similar patterns on another page that may contain fewer records,
4. Performing list structure recognition as a post-processing step to make corrections in the results of the initial list detection step,
5. Bootstrapping field dictionaries via information extraction from other lists in the same document, and
6. Acknowledging that field recognizers with a high hit-count naturally have a low standard deviation, even when they are not part of a list, by using unsupervised statistical language modeling to locate list-like patterns among the noisy labels that are unlikely to occur by chance (i.e. collocation metrics).

We expect that these ideas, especially the last four, will enable ListDetector to become more adaptive to document style and content with no added cost in terms of knowledge engineering or human supervision.

7. ACKNOWLEDGMENTS

We thank Ancestry.com for providing most of the document page images and OCR output we are assembling as part of the public corpus.

8. REFERENCES

[1] A. Belaïd. Recognition of table of contents for electronic library consulting. *International Journal on Document Analysis and Recognition*, 4:35–45, 2001.

[2] C. Chang, C. Hsu, and S. Lui. Automatic information extraction from semi-structured web pages by pattern discovery. *Decision Support Systems*, 35:129–147, 2003.

[3] N. Dalvi, R. Kumar, and M. Soliman. Automatic wrappers for large scale web extraction. *Proceedings of the VLDB Endowment*, 4:219–230, 2010.

[4] D. W. Embley, Y. S. Jiang, and Y. Ng. Record-boundary discovery in web documents. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pages 467–478, Philadelphia, Pennsylvania, USA, 1999.

[5] E. Green and M. S. Krishnamoorthy. Model-based analysis of printed tables. *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, 1072:214–217, 1995.

[6] R. Gupta and S. Sarawagi. Answering table augmentation queries from unstructured lists on the web. *Proceedings of the VLDB Endowment*, 2:289–300, 2009.

[7] J. Hu, R. Kashi, D. Lopresti, and G. Wilfong. Evaluating the performance of table processing algorithms. *International Journal on Document Analysis and Recognition*, 4:140–153, 2002.

[8] M. Hurst and T. Nasukawa. Layout and language: Integrating spatial and linguistic knowledge for layout understanding tasks. In *Proceedings of the Eighteenth Conference on Computational Linguistics*, pages 334–338, Saarbrücken, Germany, 2000.

[9] D. X. Le and G. R. Thoma. Automatically creating biomedical bibliographic records from printed volumes of old indexes. In *Proceedings of the 9th World Multiconference on Systemics, Cybernetics and Informatics*, pages 267–274, Orlando, Florida, USA, 2005.