Associating information extracted from the web into records

Adwait Ratnaparkhi

Abstract
Information extraction from web pages holds promise because of the size and richness of the internet, but also presents difficulties because the interesting facts are usually presented in an unstructured format. We use finite state and statistical techniques to extract fields of interest from web pages, such as names, addresses, and phone numbers. But ultimately, we are interested in collections of fields, or records, and therefore need to perform the record association task, in which the goal is to assemble a coherent record from an ambiguous list of candidates for each field. We wish to recover structured information records — specifically, corporate contact information such as name, address, and phone number — from unstructured web sites. We frame our problem as a graph search problem, and implement a solution with an exact $n$-best graph search technique and a heuristic weighting function. We then use this system as a bootstrapping step to train a decision tree classifier to recognize well-formed records. We present experimental results and comparisons with a baseline system.

1 Introduction
Web pages on the internet are a rich and vast supply of data for applying information extraction technologies. However, the interesting information on web pages is not guaranteed to be structured in a way that is easily readable by automatic techniques. Certain facts, like names and addresses, can usually be represented with textual strings found on the web site; we refer to such facts as fields. A collection of fields is called a record, and the task of constructing the record from many candidates for each field is called record association.

In this paper we look at finding the corporate information record, consisting of company name, street address, and phone number from a company’s web site. While some web sites of larger companies tend to have the corporate information neatly presented on a single page in the site, most small companies tend to have it scattered across the page, or even scattered across multiple pages on the site. There are usually many good candidates for a given field, and accurate record association therefore requires us to consider a combinatorially large space of possible records.

2 System Description
The system we describe starts from a list of URLs for the web sites of interest, and attempts to produce a single information record for each web site. The major components of the system are:

Crawler: Given a list of web sites, crawls a subset of their pages and stores them in a local filesystem.

Candidate Generator: Produces lists of candidates for each field. The intention is to overgenerate, so the lists have high recall but low precision. This step uses a combination of finite state techniques and statistical metrics.

Span Classifier: For each field, we train a binary classifier, and assign each candidate a score of being the correct choice for the field. This step requires labeled data to train the classifiers.

Record Associator: Given many candidates for each field and their scores, we search for the best record, which is comprised of exactly one candidate per field.

The field type tells us the attribute name of the fact, e.g., company name would be a valid field type. The field values are the strings of text found on the web site that correspond to the field types, e.g., “IBM” would be a valid field value for company name for the web site www.ibm.com. We are interested in the following field types:

Company name: The name the company uses to refer to itself in the web site.

Street Address: The main street address of the company.

Phone: The main phone number of the company.

Aspects of the candidate generation, span classification, and record association are highly specific to
the particular choice of fields, but the general architecture of our system, as well as the algorithms of the record associator are field independent. This paper briefly describes candidate generation and span classification, and is mostly concerned with record association. The description of the crawler is beyond the scope of this paper. Table 1 shows an example of an unambiguous record, with one field value per field type. In practice, we will need to decide from among many possible field values for each field type.

3 Candidate Generation

In the candidate generation phase, the system marks a number of contiguous textual strings occurring in the web site, or spans, as potential candidates for a given field type. Finite state techniques and unsupervised statistical measures are used to hypothesize usually in between 10 and 100 candidate spans; the candidate generation need not be very precise, since the intention is to overgenerate. Many candidate spans may occur several times throughout the site, so re-occurring spans are consolidated into a single field candidate. If $c_{ij}$ is the $j$th field candidate for the $i$th field, then let $s_{ijk}$ denote the $k$th span for $c_{ij}$. The field candidate $c_{ij}$ can be viewed as an equivalence\(^1\) class for the spans $s_{ij1} \ldots s_{ij|c_{ij}|}$.

4 Span Classification

A span classifier looks at the context surrounding a span — HTML formatting and text — and assigns it a weight. Unlike candidate generation, the goal is to obtain a fine-grained judgement on the acceptability of a span to represent a certain field for the web site. The span classifiers are trainable, but the details are not included here.

We derive field candidate weights by averaging over the span weights. Let $C_i$ denote the classifier for the $i$th field, and let $w_1$ denote the field candidate weight:

$$w_1(c_{ij}) = \frac{1}{|c_{ij}|} \sum_{k=1}^{|c_{ij}|} C_i(s_{ijk})$$

(1)

\(^1\)Minor variations in punctuation were considered equivalent, e.g., “FooBar Inc” and “FooBar Inc.”

The weight given by any classifier $C_i$ and $w_1$ is guaranteed to be in between 0 and 1.

5 Record Association

The goal of record association is to find the best information record given a list of candidates for each field type. If $L_i = \{c_{i1} \ldots c_{iJ}\}$ is the candidate list for the $i$th field type, the record association algorithm will take the lists for $N$ fields $L_1 \ldots L_N$ and produce a single record $x_1 \ldots x_N$, such that $x_i \in L_i$. We compare three algorithms for record association: a baseline algorithm, an algorithm which models the record association problem as a weighted graph search, and an algorithm which uses machine learning to predict the best record.

6 Model 1: Baseline Strategy

The baseline strategy is to use to decide the candidate on a per-field basis, using only the weighting function (1):

$$x_i^* = \text{argmax}_{j \in \{1 \ldots |L_i|\}} w_1(c_{ij})$$

for $0 \leq i \leq N$. With this strategy, it is not possible to capture relationships between fields that either count towards or against their association.

7 Model 2: Graph Search

The graph search model for record association is motivated by 2 observations:

- The distance on the page between two field candidates is important in determining their association
- The $n$th best field candidate from equation (1), for $n > 1$, may be a better choice for the best record instead of the first best candidate.

We first define a distance function, and then define a weighted graph search model that uses the distance function in its weighting scheme, and allows us to explore combinations of the $n$-best field candidates.
Figure 1: Graph for Model 2. Columns represent fields (name, address, phone). Rows represent different candidates for a given field. Edges have weights. Best path through graph represents best record.

7.1 Span distance

The distance between 2 spans on a web page is highly correlated with their association. A company name is more likely to be associated with a nearby address than with some other address further away on the same page, or on some different page altogether. While span distance can be computed naturally, e.g., in terms of the number of tokens separating the 2 spans, the distance between 2 field candidates is ambiguous since the spans comprising each field can have varying distances to each other. For our purposes, we define field distance $d_f$ as the minimum distance of the spans comprising the fields.

$$d_f(s_1, s_2) = \begin{cases} \text{token-distance}(s_1, s_2) & s_1 \text{ and } s_2 \text{ are on same page} \\ D & s_1 \text{ and } s_2 \text{ are on different pages} \end{cases}$$

$$d_f(c_{ij}, c_{i'j'}) = \min_{k \in \{1 \ldots |c_{ij}|\}, k' \in \{1 \ldots |c_{i'j'}|\}} d_s(s_{ijk}, s_{i'k'})$$

The choice for $D$ is made heuristically, we use $D = 10000$ for our experiments.

7.2 Weighted Graph Search

Given a weighting function based on distance, we define the best record as the one which maximizes the pair-wise weights between its fields. Given $N$ fields and $J$ candidates per field on average, a brute force approach would enumerate $J^N$ records, compute the pair-wise score between candidates of different fields for all records, and choose the one record with the highest score.

In order to avoid such a combinatorially expensive method, we make the simplifying (markov-like) assumption that the fields are inherently ordered, in that the placement of field $i + 1$ only depends on the placement of field $i$. We therefore need only compute the pair-wise weights of between candidates of "adjacent" fields. The total score $w_2$ of a record $x_1 \ldots x_N$ is then:

$$w_2(x_1 \ldots x_N) = \prod_{i=1}^{N-1} g(x_i, x_{i+1})$$

$$g(x, y) = \frac{1}{1 + d_f(x, y)}$$

The weight function $w_2$ is for records, while $g$ is for field candidate pairs. One can imagine many simple functions which inversely weight the distance; $g$ was chosen based on its performance on a small set of examples.

The problem now is to search all possible records, i.e., sequences $x_1 \ldots x_N$, for the one which maximizes $w_2$. We can do this efficiently if we model the record search as a search for the highest scoring path through a weighted directed graph. Given $N$ field types, the vertices of the graph can be arranged in $N$ columns. The $i$th column represents the field candidates for the $i$th field, so $c_{ij}$ will be in the $j$th row of the $i$th column. Every candidate from the $i$th column has a directed edge to every candidate in the $i + 1$st column, i.e., there are edges $c_{ij} \rightarrow c_{(i+1)j'}$ for all $j, j'$ and $i < N - 1$. We add dummy start and end states with 0 weights for the sake of convenience. The best path through the graph corresponds to the record with the highest score according to $w_2$.

$$x_1^* \ldots x_N^* = \arg \max_{x \in L, 0 \leq i \leq N} w_2(x_1 \ldots x_N)$$

An example graph is shown in Figure 1. Our problem-specific field type ordering is as follows: company name is field 1, postal address is field 2, and phone number is field 3. Finding the best path is accomplished easily with dynamic programming (aka Viterbi algorithm). We prune the graph before finding the best path by keeping only the top $K$ (=5) candidates — as ranked by equation 1 — for each field. This prunes out many of the noisy candidates which otherwise may score well with the distance-based weighting function. We also apply the $n$-best graph algorithm described in (Tran et al., 1996) to the pruned graph, which gives us the exact $n$ best paths through the graph with respect to the weighting function. The $n$ best records corresponding to these paths are useful for Model 3, described below.

7.3 Model 3: Machine Learning

The machine learning model for record association is motivated by the observation that while distance appears to be the most obvious correlate with association, there may be other features that are important but too subtle and complex to explicitly capture with a manually written pattern. A machine learning algorithm, such as a decision tree, could induce complex features across candidates from different fields, and may capture generalities not present
in Models 1 and 2. We can frame record association as a binary classification problem: the machine learning algorithm, once trained, will take a record \((x_1 \ldots x_N)\) and predict either \textit{true} or \textit{false} (with confidence values or probabilities), corresponding to a well-formed or ill-formed record, respectively.

### 7.3.1 Synthetic Training Data

Training a learning algorithm for this task requires a list of labeled examples, where each example has the form \(\{l, x_1 \ldots x_N\}\), such that \(x_i\) is a candidate for the \(i\)th field type, and where \(l \in \{\text{true}, \text{false}\}\) is a label. However, such data was not readily available. We did have data that was used to train the span classifiers (section 4), but using it to generate record level training data presented some problems.

For example, the labeling was at the field level and not at the record level; it only specified which field candidates were correct, i.e., it consisted of examples of the form \(\{l, c_{ij}\}\), where \(c_{ij}\) is the \(j\)th candidate for the \(i\)th field. In the case where exactly one candidate from each field is marked as \textit{true}, the positive record level training instance is easy to obtain — it is the one comprised of only the positively-marked candidates from each field. In cases where there were multiple candidates marked \textit{true} for a given field, choosing a correct record is ambiguous. And sometimes no candidates were marked as \textit{true}, in which case we cannot hope to retrieve a positive training instance. But the most problematic issue was the generation of negative record level training instances, since we needed to define what it meant to be a negative record, and also how to choose the best negative training instance from the many that could conceivably be generated from the existing field-level labeled data.

We resolved these issues by creating \textit{synthetic} training data from Model 2 as follows:

**Positive Training Instances:** We define a positive record \(x_1 \ldots x_N\) as one in which all of the field candidates \(x_i\) have been marked with \textit{true}. We used Model 2, but instead of constructing a graph from the top \(K\) positive and negative field-level instances, we constructed a graph from only positive field-level instances, using at most \(K\) for each field. The record derived from the best path in the graph would therefore be guaranteed to contain only positively marked fields. For cases where only one candidate was marked positive for field, the graph search is unnecessary, since the positive training instance can be derived by taking all the positively marked field candidates. For cases where multiple candidates were marked positive for one or more fields, the best path gives us the record that maximizes \(w_1\) over the space of positive records.

**Negative Training Instances:** We define a negative record \(x_1 \ldots x_N\) as one in which at least one \(x_i\) is marked with \textit{false}. Given the graph structure in Model 2, the \(N\)-best algorithm efficiently gives us the \(N\) highest scoring records. We scan through the list of records, and use the highest scoring negative record as our negative training instance. Our hope is that the negative training instance will be a good “imposter”, in that it is ranked highly by Model 2 yet contains negatively marked fields.

### 7.3.2 Decision Tree Model

The positive and negative training instances are used to train a decision tree model with the features listed below. We used decision trees because of their feature induction capability; we are especially interested in inducing features that express arbitrary relationships between two or more fields, since Model 1 did not express any intra-field relationships, and Model 2 could only express relationships between two adjacent fields, using only distance.

The decision tree model is an implementation of (Quinlan, 1992) which handles real-valued features. The features used in the model are:

- Distance-based weight score for record: use weighting function \(w_2\).
- Field confidence for all fields in record: use weighting function \(w_1\) on all fields.
- Token distance between all adjacent fields in record: use distance function \(d_f\) on adjacent fields.

We experimented with the following features but they did not yield any improvement:

- XPath prefix distance between adjacent fields in record: this is an alternative to token distance. XPath is a language for identifying elements in Xhtml by their paths from the document’s root. If two spans share a large fraction of their XPath prefix, they are likely to be in structurally similar positions.
- Check if the company name is in the web site’s title: even if the token distance is great, being in the title may count positively towards association.

The weight given by the decision classifier is the probability that the record of interest has a \textit{true} label, i.e., it is a well-formed record.

\[
w_3(x_1 \ldots x_N) = P(\text{true}|x_1 \ldots x_N)
\]

The decision tree is applied to the top \(M\) (=20) records obtained from applying the algorithm of
(Tran et al., 1996) to Model 2, and the highest scoring record is returned as the answer.

\[ x_1 \ldots x_N = \operatorname{argmax}_{x_1 \ldots x_N} \in \text{top}(M, w) \ w_3(x_1 \ldots x_N) \]

\[ \text{top}(M, w) = \text{the } M \text{ highest records according to weight function } w \]

8 Evaluation

We evaluate models 1, 2, and 3 by running our entire system on a random sample of web sites representing small companies. We only evaluated our systems on records for which there existed a company name, an address, and a phone number.\(^2\) Furthermore, we discarded the record if any of its fields fell below an empty field threshold. The threshold was used to prune out cases where a record was successfully found, but where all of its chosen field candidates were of poor quality, and had low field-level scores. The systems are evaluated on precision and recall, defined as follows:

\[ X = \text{number of sites for which we correctly guess the company name, address, and phone} \]

\[ Y = \text{number of sites for which we guessed a company name, address, and phone} \]

\[ Z = \text{number of sites for which there existed a company name, address, and phone} \]

\[ \text{Precision} = \frac{X}{Y} \]

\[ \text{Recall} = \frac{X}{Z} \]

Three non-overlapping sets of crawled web sites were used in the evaluation: a training set of 254 sites, for training the decision tree record classifier and the span classifiers, a development set of 104 sites, for evaluating and tuning the various parameters and thresholds, and an evaluation set of 73 sites, for reporting results. The results are shown in Table 2. The empty field threshold can be used to trade off precision and recall; higher values will tend to increase precision, since the fields which exceed this threshold are more likely to be correct. All systems in Table 2 have been tuned to produce roughly 90% precision\(^3\) (or, as close as possible to 90% precision), so the real comparisons are in the recall numbers. Model 2, the graph search model using a distance-based weighting function, improves over the default Model 1, while Model 3, the decision tree record classifier, improves over Model 2. Furthermore, these improvements are consistent over both development and evaluation sets.

For some sites, our evaluation framework allows multiple records to be counted as correct. Because our test sets do not have record-level annotation, we are forced to use an ambiguous definition for record “correctness”. It is usually the company name that creates the ambiguity, e.g., “FooBar”, “FooBar, Inc.” and “FooBar.com” might all be acceptable variants of a company’s name. If all of the above variants have been labeled with true, a record with any of the variants above, as well as correct entries for the other fields, would be marked as correct in our evaluation scheme.

9 Discussion

Typically, most IE applications find fields from documents, and use heuristics – usually based on intra-field distance — to associate fields together. While Model 2 shows that distance is an important factor, combining the confidences of previous weighting schemes, as done in Model 3 with machine learning, can lead to even greater improvement. Machine learning provides a benefit in this domain because our attempts at hand-coding heuristics to improve precision and recall (beyond Model 2) were unsuccessful; most of the easily observed features were site-specific and would therefore not generalize to new sites.

Training a decision tree for Model 3 was not straightforward due to the lack of annotated data. We overcame this issue by essentially bootstrapping from Model 2, and training the decision tree with synthetic data. In particular, it was not clear how best to obtain negatively labeled training data, and the strategy described in section 7.3.1 was absolutely essential to making Model 3 work.

Our system is especially useful for unstructured web sites. In large, professionally created, structured sites, the contact information is usually presented in a way that allows easy extraction. In smaller, less professional sites, the information is scattered throughout the site, in places such as the title, sidebars, headings, tables, and paragraph text. The diversity of places in which the information requires some kind of non-trivial record association strategy.

Currently, our features for Model 3 consist of intra-token distance, per-field scores, and the distance-based score from Model 2. Using additional features about the page layout have not seemed to help yet, but we believe that there are still rich opportunities for using such features. Right now our distance function assigns an arbitrary large constant to spans on different pages; we are still investigating ways to measure distance of spans on different pages that correlate with association. Ultimately, the goal of our systems is to reverse-engineer the visual and textual cues employed by the web site designer to

\(^2\) for internal reasons too specific to be discussed here, we were not interested in companies that didn’t satisfy this criterion.

\(^3\) for internal reasons, we were not interested in data below 90% precision.
<table>
<thead>
<tr>
<th>Model</th>
<th>Development (104 sites)</th>
<th>Evaluation (73 sites)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>%Precision</td>
<td>%Recall</td>
</tr>
<tr>
<td>Model 1</td>
<td>88.1</td>
<td>51.5</td>
</tr>
<tr>
<td>Model 2</td>
<td>93.3</td>
<td>56</td>
</tr>
<tr>
<td>Model 3</td>
<td>91.4</td>
<td>64</td>
</tr>
</tbody>
</table>

Table 2: Evaluation of systems on web sites. All models are tuned to a precision as close as possible to 90%.

present the fields of the record. As these cues are discovered, we can use them as features in Model 3.

Much of the architecture of Models 2 and 3 can be used for other information extraction tasks. The graph search technique and distance-based weighting function of Model 2, as well as the use of Model 2 to bootstrap Model 3, are independent of the type of information we are extracting. The features for the decision tree of Model 3, the span classifiers, and the finite state techniques used to generate candidates are specific to our problem. The ordering imposed on the field types is also problem-specific and must be chosen by the experimenter. Information extraction applications that need to collect fields scattered across a document into a single record can use this architecture. While our system produces one record per site, it can be extended to produce multiple records per site by running it multiple times on a site with carefully chosen candidate lists on each run.

Our work is related to work on wrapper induction, such as (Soderland, 1997) and (Kushmerick et al., 1997). These papers are mainly concerned about learning the wide variety of HTML formatting cues used to present facts on web pages, so that they know where to look when extracting fields from web pages. We assume that candidates for fields have been generated in advance (in our case, by the candidate generator and span classifier) and use our record association algorithm to explore various ways of putting the information together. While our span classifier considers formatting cues in assigning weights, our record association algorithm relies very little on HTML formatting cues, which allows it to work with unstructured text as well as structured text.

10 Conclusion

This paper presents three systems for associating extracted fields into records. Model 2 confirms that proximity is positively correlated with association by outperforming the baseline Model 1, while Model 3 shows that machine learning can improve performance over using distance alone. Models 2 and 3 make no assumptions about the structure of the web page, and are especially useful on sites in which the information is scattered across one or more pages. We believe our architecture is generic enough to perform record association for most other text and web related information extraction problems.

11 Acknowledgments

This work was performed in the spring of 2002 when the author was employed by WhizBang! Labs, 4616 Henry Street, Pittsburgh, PA 15213.

References


Bach-Hiep Tran, Frank Seide, and Volker Steinbiss. 1996. A Word Graph Based N-Best Search in Continuous Speech Recognition. In The Fourth International Conference on Spoken Language Processing (ICSLP), Philadelphia, PA.