Learning Knowledge Bases for Information Extraction from Multiple Text Based Web Sites

Xiaoying Gao and Mengjie Zhang
School of Mathematical and Computing Sciences
Victoria University of Wellington
PO Box 600, New Zealand
{xgao,mengjie}@mcs.vuw.ac.nz

Abstract

We describe a learning approach to automatically building knowledge bases for information extraction from multiple text based web pages. A frame based representation is introduced to represent domain knowledge as knowledge unit frames. A frame learning algorithm is developed to automatically learn knowledge unit frames from training examples. Some training examples can be obtained by automatically parsing a number of tabular web pages in the same domain, which greatly reduced the time consuming manual work. This approach was investigated on ten web sites of real estate advertisements and car advertisements and nearly all the information was successfully extracted with very few false alarms. These results suggest that both the knowledge unit frame representation and the frame learning algorithm work well, domain specific knowledge base can be learned from training examples, and the domain specific knowledge base can be used for information extraction from flexible text-based semi-structured Web pages on multiple Web sites.

Keywords: Information extraction; learning; knowledge unit frame; text-based web sites; semi-structured data.

1. Introduction

Information extraction from Web is a difficult task. More and more online documents are becoming available and each has a different data format. The number of Web sites and their domains is huge and is growing very fast. The existing Web pages are continuously being updated and their data formats may be changed or updated at any time without warning. While it might be easy to hand craft an information extraction system for one particular Web site in one specific domain at a particular time, how to maintain the system and make it effective, efficient, and adaptive for multiple or new Web sites is a big challenge. There is an urgent need to develop methods and tools to ease the system generation and adaptation.

Recently, a number of information extraction systems have been developed on semi-structured web pages, including manual [7], semi-automatic [2, 3], interactive [1, 11] and automatic [5, 8] systems. However, the wrappers generated by these systems are site specific, that is, an individual wrapper has to be built for each Web site and could not be applied to other Web pages/sites in the same domain.

In addition, most of these systems are developed and applied to Web pages with a strict data format. In Doorenbos [5], it is assumed that data are presented in a uniform format and related data are presented in one single line. The algorithm introduced in Kushmerick [8] works on Web pages where data are presented in a fixed order without any missing items. The system by Ashish [3] is based on heuristics on font size and indentation and works on pages organized hierarchically. More flexible semi-structured Web sites such as those with free text, with missing units or with free order units are beyond the scope of such systems.

The goal of this paper is to develop a learning/adaptive approach to automatically building a knowledge base for information extraction. Rather than being applied to relatively structured tabular web pages, this approach is designed for information extraction from more flexible, relatively unstructured text-based Web pages such as pages with missing data or data written in free order or even in free text. Figure 1 shows an example of the kind. Instead of building site specific wrappers, this approach will build a single wrapper that can work on multiple Web sites in the same domain. To investigate the effectiveness and adaptability of this approach, flexible text based web sites such as classified real estate advertisements and car advertisements are used as the test beds in the experiments. We also hypothesize that domain knowledge (and common sense knowledge) is important for guiding information extraction from text based
ATTENTION TENANTS
RINGWOOD $105 pw
Neat large 1 BR flat, own yard.
HEATHMONT $140 pw
Neat 2 BR unit, g'htg, lge yard.
Elders Ringwood 9870 0111
118 Maroondah Hwy, R’wood

ALBERT PARK $500 PW
Large 3 bedroom pavilion, formal and informal dining areas,
2 bathrooms, study, good size.

Figure 1. A Fragment of a Web page. (a) Postscript format; (b) HTML format.

The approach mainly consists of four parts: training example collection, knowledge representation, a learning system to automatically learn and construct a knowledge base, and an inference engine (a matching system) which applies the learned knowledge base to the new unseen text based web pages to extract data as the final results. This paper mainly focuses on knowledge representation and the knowledge unit frame learning algorithm.

2. Knowledge Representation

2.1. Knowledge Unit Frames

For convenience, we call the extracted information as knowledge units, denoted as ku(Name, Value). For example, for the first advertisement in figure 1, the knowledge units to be extracted are: suburb, price, size, and type, denoted by ku(suburb, ringwood), ku(price, 105), ku(size, 1), and ku(type, flat). Each knowledge unit has a name and a value. Its name suggests its content, which is the meaning of the data. Its value is the information to be extracted.

In this approach, we use knowledge unit frames to represent the domain knowledge that is useful for extracting knowledge units. The BNF format of our knowledge unit frames is as follows:

KnowledgeUnitFrame ::= FrameList
FrameList ::= Frame | FrameList
Frame ::= SlotList
SlotList ::= Slot | SlotList
Slot ::= OneSlot OR SubFrame
OneSlot ::= (SlotName, Slot Value)
SubFrame ::= Frame

As shown in figure 2, some important features that are useful for knowledge unit recognition and value extraction are chosen to form the slots. Each frame can have 2 or 3 slots (the second slot is optional) and a set of sub frames. The number of lexical items of a frame defines the number of sub frames — each lexical item is represented as a sub frame. The value extraction function defines how to extract the value of the knowledge unit. The certainty factor is used to provide a criterion for choosing between frames. The details of the value extraction function and the certainty factor will be discussed later in this section.

Each subframe has 2 to 8 slots. A subframe must include the second slot style and one of the third slot instances and the fifth slot pattern. All the other slots are optional. The value slot specifies the value returned (extracted) by this subframe. The default value is the extracted string when the slot is omitted. If the value is specified, then this subframe will return the specific value. For example, suppose a subframe can extract strings such as “flats”, “flat”, “SC FLAT”, “Apartment”, “apart”. If we would like the value
of the lexical item to be “flat”, then we specify this slot as value(‘‘flat’’). If the value slot is omitted, any of the extracted strings such as “flats”, “SC Flat” can be returned as the value. The style slot specifies the data type of the lexical item, including tag, character, word, phrase, digit, number and string. The tag here means HTML tags, which can be used to define lexical items in our system. The instances slot contains positive keywords in a list, suggesting that the knowledge unit is present. The exceptions slot contains a list of negative keywords, suggesting that the knowledge unit is not present. The lexical item extraction pattern slot can have many patterns. Each pattern has two sub slots: a pattern function to identify the pattern values, for example, any_number(X) for extracting “123” and “5,000”, and a certainty factor to specify the priority of the pattern. The pattern function and the certainty factor will be detailed later in this section. The max length and min length slots specify string length constraints, and the min and max slots specify range constraints for numeric lexical items.

2.2. The Value Extraction Function

In this representation, each subframe extracts a lexical item from a knowledge unit example and all lexical items are returned as a vector \([v(1), v(2), \ldots, v(n)]\), where \(v(i)\) is the value extracted by subframe \(i\). The value extraction function defines how to work out the value of the knowledge unit from the vector. For example, multiple(ItemList) returns all the items in ItemList. For example, the string “2-3 Bedrooms” for Size returns vector \([2, 3, \ldots, \text{“bedrooms”}]\). If we want both the value “2” and “3”, we can define the value function as multiple(\([v(1), v(3)]\)).

2.3. The Pattern Function

Our system supports 11 pattern functions. any_string matches anything; any_tag matches anything between < and >; any_number matches all strings with any digits including digits containing ‘‘, ’’ in between such as ‘‘2,000’’; any_word matches all strings with any alphabetic character and characters with ‘‘/’, ‘‘. ’’ in between; any_phrase matches any word with only spaces in between; any_word_in_capital matches any word printed in capital letters; any_phrase_in_capital matches any phrase printed in all capital letters; any_delimiter matches any punctuation and special characters; any_digit matches any digit; and any_char matches any character. In addition, a special pattern function is defined as match_string_in_db(DatabaseName), which matches a string in a database. For example, match_string_in_db(suburb) can be used to extract a particular suburb from a suburb database.

2.4. Certainty Factors

Certainty Factor (CF) is a number between 0 and 1. The frame or pattern with a higher CF has a higher priority, that is, when there are more than one frame or pattern to be used, the one with higher priority is triggered first. The patterns which are often used and make fewer mistakes have a higher CF, while not commonly used or possibly ambiguous ones have a lower CF. The CF value is calculated during the frame learning process, which will be detailed later.

2.5. An Example

The representation of knowledge unit Price is given as an example, as shown in figure 3. This frame is expressive enough to extract price from “$120 PW”, “800PW”, “$1,200/pw”, “$300 p/w”, or “420 PWeek”. Similarly, we use other frames to extract price from “Price: 800”, or “£700 per month”. In our system, price is represented in five frames.

<table>
<thead>
<tr>
<th>Frame: price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledge Unit Frame NO. 1</td>
</tr>
<tr>
<td>No of lexical items: 4</td>
</tr>
<tr>
<td>Value extraction function: single(v(2))</td>
</tr>
<tr>
<td>Certainty factor: 1.0</td>
</tr>
<tr>
<td>Sub Frame No.1</td>
</tr>
<tr>
<td>Style: char</td>
</tr>
<tr>
<td>Instances: [“$,”]</td>
</tr>
<tr>
<td>Pattern No.: 1</td>
</tr>
<tr>
<td>Certainty factor: 1.0</td>
</tr>
<tr>
<td>Max: 9999</td>
</tr>
<tr>
<td>Min: 0</td>
</tr>
<tr>
<td>Sub Frame No.3</td>
</tr>
<tr>
<td>Style: char</td>
</tr>
<tr>
<td>Instances: [“p/w”, “per week”, “pwe”, “p/week”, “p/week”]</td>
</tr>
</tbody>
</table>

Figure 3. An Example of Knowledge Unit Frames

3. Algorithm for Learning Knowledge Unit Frames

To avoid manually writing knowledge unit frames or at least reduce the amount of manual work, we developed a frame learning algorithm. The inputs of this algorithm are knowledge unit examples. If an example is positive to a particular knowledge unit, it would be a negative example to other knowledge units. The outputs of this algorithm are the frames for these knowledge units. The algorithm is shown as follows:

1. For the positive examples of each knowledge unit, do the following training process:
(a) Initialize the first example as a tree. The name of the knowledge unit is the root, and each lexical item is a leave.

(b) For each new example, initialise it as a tree, then try to combine this tree with the previous tree by
   - Merging nodes if the nodes have the same meaning, or
   - Grouping nodes via creating a generalized node

(c) Stop when all positive examples are exhausted.

2. Validate the learning results on both positive and negative examples.

3. Write the learned trees to frames and save frames into the knowledge base.

Figure 4 shows an examples of learning a price frame from two price examples $200 PW and $600p/week. The algorithm is detailed in the rest of this section.

3.1. Initialising Trees

Each knowledge unit example is initialised as a tree. We have five basic data types: phrase, word, number, special character and punctuation (delimiters). The input string is usually taken as a phrase. It is split when it has a number or a special character such as $, &, @, or space. During the splitting, a phrase is split to 4 other types of data. For example “$200 PW” is initialized to a tree with four sub nodes “$”, “200”, “ ”(space), “PW”.

3.2. Merging Nodes

Two nodes a and b will be merged if one of the four following conditions is met: (1) a and b are the same by ignoring letter cases, e.g. “FLAT” and “Flat”; (2) a and b are based on the same word root, e.g. “Flat” and “Flats”; (3) a and b are synonyms, e.g. “Flat” and “apartment”; and (4) a and b are abbreviations of the same word or phrase, or one is the abbreviation of another (e.g. “apartments”, “apart”, “aparts”). In addition, as a special case, any node a can merge with empty node φ.

Of all the four above considerations, the first is straightforward. While natural language processing techniques [9, 10] might be quite helpful for the other conditions, there will be a big speed and memory cost. Considering the fact that the number of words with the same root and the number of synonyms are relatively small in our test domains, we hand-coded a small dictionary to meet our goals for the second and the third conditions. To detect whether two words or phrases a and b are abbreviations, we defined the following two rules:

\[
\text{if } a \text{ is a child of } b \text{ (or } b \text{ is a child of } a \text{), then } a \text{ and } b \text{ are abbreviations.}
\]

3.3. Grouping Nodes by Generalization

We use a generalisation-tree to represent different types of strings, as shown in figure 5. The top level is any string. The bottom level is specific data with a particular type such as phrase, special characters, and numbers. Each type of data in the bottom level can be generalized by going upward the tree. We expect the tree to be completely domain independent and site independent so that the learning algorithm is general enough to work on any Web site in different domains.

In order to make the learning effective, we limit the scope of generalization by introducing length constraints for words and phrases and range constraints for numbers. Two parameters MaxLength and MinLength are used to record the maximum and minimum length for words and phrases. Similarly, Max and Min are used to record the upper and lower boundaries of numbers.

To reduce overgeneralization, we use Certainty Factor (CF) [12] to evaluate the level of generalization. Bottom level nodes are specific data and the CF for each node is set to 1. The top level node can be matched with any input so the CF is set to 0. The CFs of the nodes in the middle levels are between 0 and 1, for example, any phrase and any word have very low CFs since they can almost match any input within the length range; any number and any char have a higher CF since they are also limited in data types. The CF values for each data type in the tree are predefined, as shown in figure 5.

CF is used to control the level of generalization. When a node is generalized, its CF becomes lower. The CF of a tree is the maximum CF of all sub nodes. If the CF of the tree is greater than a threshold ε, the generalization will be accepted and accordingly the two examples forms a single tree.

Otherwise, the generalization is rejected and two trees need to be formed. Through an empirical search through experiments, we chose ε = 0.5. Let us examine the price example.

The second node in figure 4 in both training examples “$200 PW” and “$600 PW” can be grouped together by creating a generalised node any_number(max=600, min=200).

The CF of the generalized node is 0.8 (see figure 5), which is lower than either of the CFs of the 2 nodes (both of them are 1.0). The CF of the tree now is:
Learning one frame of Price from two examples "$200 PW" and "$600p/week"

(1) The first example "$200 PW"
(2) A new example "$600p/week"
(3) Merging "$" and "$", "PW" and "p/week", and "space" and ""
(4) Generalising to group "$200" and "$600"
(5) Validating:
If we only have the two examples, then
DP=2, TP=2, FP=0, TP/(TP+FP)>threshold
(6) Writing to frame:
Each box is a sub tree and is written as a sub frame.

Figure 4. An Example of Learning KU Frames from Training Examples

\[ CF(tree) = max\{CF(\$), CF(any\_number), CF(space), CF(PW)\} = max\{1, 0.8, 1, 1\} = 1 > \varepsilon \]
So the generalization is accepted. It is important to note that we used the maximum rather than minimum in the tree generalisation since the maximum is suitable for our purpose, that is, as long as one of segments/items in a generalised pattern (associated with a tree) is confident (>0.5), the pattern would be valid and accepted.

### 3.4. Validating

In the training process described above, only positive examples were used. For unseen web pages, it is quite possible for negative examples to be incorrectly considered as positive passing through the trees, which often causes problems. To avoid these problems, we use both positive and negative examples as a tuning set to validate generalized trees.

Assuming \( DP \) is the total number of positive examples for a generalised tree, \( TP \) (true positives) is the number of positive examples correctly passing through the tree, and \( FP \) (false positives) is the number of negative examples incorrectly acting as positive examples passing through the tree, a generalized node will be rejected if

\[
FP > 0 \quad \text{and} \quad \frac{TP}{DP} < \varepsilon_1
\]

or

\[
\frac{TP}{TP + FP} < \varepsilon_2
\]

In this research, we chose \( \varepsilon_1 = \varepsilon_2 = 0.5 \) based on our experiments.

### 3.5. Writing Trees as Frames into Knowledge Base

In this step, each tree is written as a frame and saved into the knowledge base. Each node, either a merged node, generalised node, or an isolated node, which represents a sub tree, is written into a sub frame.
4. Results and Discussions

In this research, we did two groups of experiments in the domains of real estate advertisements and car advertisements. After briefly describing training examples used for the two experiments and the performance measurement, this section presents the final information extraction results.

4.1. Training Examples

Manually collecting sufficient training examples of domain knowledge is very time and cost consuming. In this approach, besides manually collecting some examples from a single web page of a site, we also use the knowledge units automatically extracted by our former wrapper [6] from tabular web pages in the same domain as training examples. While selecting such kinds of tabular web sites needs a bit of manual searching work, it would still reduce the total amount of manual work.

Ideally we can collect sufficient training examples from the corresponding tabular web pages only, however it is not practical within a limited time. In the experiments, we used 3 tabular web sites for the real estate advertisements and 4 sites for the car advertisements for the experiments from which our former tabular wrapper [6] extract knowledge units as training examples.

4.2. Performance Evaluation

In this approach, we use recall and precision to measure the system performance. Recall refers to the number of knowledge units correctly extracted from a number of Web pages in a particular Web site as the percentage of the total number of knowledge units in these Web pages. Precision refers to the number of knowledge units correctly extracted from a number of web pages as a percentage of the total number of knowledge units extracted from those web pages.

4.3. Experiment 1: Result on Basic Corpus

The first experiment is carried out on 10 web sites in the basic corpus, consisting of 5 sites in real estate advertisement domain and 5 in car advertisement domain. The system was tested on two Web pages randomly chosen and downloaded from each site. The final results are shown in Table 1.

As shown in table 1, the performance on these text based web sites varies with the flexibility and complexity of data presentation. However, the approach achieved very promising performance: the precision and recall for knowledge unit extraction are very good (close to the ideal case) on most web sites for both domains. The system with a knowledge base for each of the two domains works well on multiple Web sites. These results suggest that this approach is suitable for information extraction from flexible text based Web sites and that it can be adapt from one domain to another by changing training examples to form the domain knowledge base. In addition, tabular web sites could be used as an important source for automatically extracting knowledge units as training examples and this would reduce the amount of manual work.

4.4. Experiment 2: Results on Two More Flexible Web Sites with More Web Pages

To investigate the extensive power of this approach, we applied our approach on more web pages on two flexible web sites. In most pages of Web sites No. 1 and No. 2 shown in table 1, data are presented as paragraphs written in almost free text. The knowledge units appeared in different order and some extra information (such as agent address and contact details) was also presented. Furthermore, a single paragraph sometimes consists of advertisements for different properties, which needs to be recognised and classified into the correct groups. While the first experiment was only performed on two pages randomly chosen, this experiment used all the 10 pages for each web sites as the test bed. The results of this experiment are shown in Table 2.

Table 1. Results for the basic testing corpus

<table>
<thead>
<tr>
<th>No</th>
<th>Domain</th>
<th>Name of Web Site</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Real</td>
<td>NewsClassifieds</td>
<td>94</td>
<td>93</td>
</tr>
<tr>
<td>2</td>
<td>Estate</td>
<td>Fairfax@Market</td>
<td>98</td>
<td>94</td>
</tr>
<tr>
<td>3</td>
<td>Estate</td>
<td>Melbourne Trading Post</td>
<td>90</td>
<td>76</td>
</tr>
<tr>
<td>4</td>
<td>Ads</td>
<td>VUT University Accommodation</td>
<td>98</td>
<td>99</td>
</tr>
<tr>
<td>5</td>
<td>Infoseek: Classified2000</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Car</td>
<td>Infoseek: Classified2000</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>7</td>
<td>Car</td>
<td>Auto Trader</td>
<td>92</td>
<td>95</td>
</tr>
<tr>
<td>8</td>
<td>Car</td>
<td>Calling All Cars</td>
<td>98</td>
<td>98</td>
</tr>
<tr>
<td>9</td>
<td>Ads</td>
<td>Melbourne Trading Post</td>
<td>96</td>
<td>98</td>
</tr>
<tr>
<td>10</td>
<td>Car</td>
<td>Fairfax@Market</td>
<td>96</td>
<td>98</td>
</tr>
</tbody>
</table>

Table 2. Results from more flexible web sites

<table>
<thead>
<tr>
<th>No</th>
<th>Web Sites</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NewsClassifieds</td>
<td>95</td>
<td>95</td>
</tr>
<tr>
<td>2</td>
<td>Fairfax@Market</td>
<td>93</td>
<td>93</td>
</tr>
</tbody>
</table>

As can be seen from table 2, these results show a similar pattern to those in the first experiment. The knowledge unit frames learned from the training examples were successfully applied to extracting knowledge units from flexible text based semi-structured web pages, even for free text. This suggests that our knowledge representation and frame learning algorithm are sufficiently powerful for information formation from this kind of web pages.
5. Conclusions

The goal of this paper is to investigate a learning/adaptive approach to automatically building knowledge bases for information extraction from multiple text based web pages. The goal was achieved by developing a frame based knowledge representation and a frame learning algorithm. This approach was tested on ten web sites of real estate advertisements and car advertisements, and nearly all the knowledge units were successfully extracted with very few false alarms. These results suggest that both the knowledge unit frame representation and the frame learning algorithm work well, domain knowledge base can be learned from training examples, and the domain specific knowledge base can be used for information extraction from flexible text-based semi-structured Web pages on multiple Web sites. Some training examples were automatically collected from a number of tabular web pages in the same domain, which greatly reduced the manual work.

Our System has the following characteristics:

- Rather than using grammar patterns and linguistics rules which are commonly used in information extraction from natural language, a very small amount of domain knowledge is used to guide information extraction, which greatly reduced the complexity of knowledge representation and learning algorithm and improved the efficiency of information extraction.

- Domain knowledge is clearly separated from other parts of the system, so this system could be adapted to other domains by changing the training examples to automatically learn a domain knowledge base.

- Unlike most systems developed in this area, which usually designed for a particular web site, our system can be used for information extraction from multiple Web sites in a specific domain.

This approach has a number of limitations, which need to work out in the future:

- Although the knowledge base can be almost automatically learned by the frame learning algorithm, a bit of manual work is need to collect sufficient training examples and to check whether there are big mistakes in the learned knowledge base.

- The knowledge unit frame representation is not particularly effective in describing knowledge units with infinite numbers of values and without clear keywords and format, such as Paper Title, Book Name, Journal Name. The identification of these knowledge units is currently based on their context. Better representations of these kinds of knowledge units need to be discovered in the future.

References


