8 Hierarchical Concepts Description and Extraction

In the last section we introduced some simple extracting techniques used in the practical information extraction system CapturePlus. In this section we will introduce a new approach for feature extracting: hierarchical concept extracting. This approach is used for some complicated contents, which are difficult to extract using simple techniques. The consideration to develop this new approach is discussed. A learning algorithm for automatic generating hierarchical concept is presented in this section too.

8.1 Observation

Problems we have to faced in IE in real applications showed that incremental hierarchical concept learning is necessary for advanced IE. Observation and discussion about this topic are carried out in this section.

Traditional Information Extraction systems try to use rules or NLP algorithm to extract contents. Rules (such as regular expressions) can find pieces of fine information. However, they are not able to extract some complicate contents. Hence, while an individual date may be easy to extract, the sequence of events spread across a long narrative of news articles may be very hard to extract.

In the case study discussed in Section 4.5, if we want to extract the distance information for hiking activity, we have to face some problems, which are not solved with simple extracting techniques. At first, the distance descriptions are various. In the example (Section 4.5) we see the variation either in distance value (such as digit number “2”, number presented as word “three”, fussy description “dozens of” etc.) or in distance unit (such as km, hours, days etc.). If extraction rules (e.g. token based rule or regular expressions) are created for this description, all possible combinations of value and distance unit have to be considered. That is, if there are $m$ variations for distance value and $n$ variations for distance unit, the number of rules for extraction all possible distance description is then $m \times n$. This fact means not only time consumption for creating rules in the modelling phase, but also a slowing down in extracting phase. Furthermore, this variation causes difficulty for training. Because the learning algorithm for token-based rules needs strong evidence to ensure that rules have high relevance. That means, more examples are needed for training. Assumed we need at least 5 examples for each rule, the minimal required number of examples is calculated then as $5 \times m \times n$. For instance, in the concrete case mentioned above, there are at least 15 ways to describe distance value and 6 different distance units. The number of rules are then 90
and the required minimal examples must be more than 450. In this case, manually creating examples becomes laborious. Moreover, variation of values causes also weakness by learning. The learning algorithm could generate some rules, which are irrelevant to the trained contents. Such irrelevant rules bring many false results and impair therefore the precision. One solution to this problem is reduction of the variation. Hence, before training, we have to pre-process the input example and replace the different value with a unified form.

Another consideration to apply hierarchical concept extraction is based on the observation that a complex concept is made up of some simpler concepts. For instance, given example as description about “hiking”. We can describe the sentence top-down as following: at first, the sentence consists of hiking information and distance information. Again, hiking information can be extracted by synonym collector with synonyms “hiking”, “hiker”, “hikers” etc. Distance information can be decomposed as distance value and distance unit. By analogy, distance value can be extracted by token based rules and some synonyms. If we construct concepts contrariwise to the above step, we can at first use token based rules and synonym thesaurus to train “distance value” (denoted as <DIS_VALUE>, for example) and “distance unit” (denoted as <DIS_UNIT> as example). Then, we can contribute the concept “distance” as “<DIS_VALUE><DIS_UNIT>”, without the need of detail knowledge about the concept “distance value” and “distance unit”. Intuitively we can contribute more complicated concept using this methods.

In this hierarchical way, theoretically, the whole document can be described as following: at first some isolated, primitive concepts, then composed concepts based on such simple concepts, then sentences based on some complicated concepts, the paragraph and finally the topics.

The new idea of hierarchical concept description and extraction consists of then two consideration: at first we replace occurrences of lower level concept with formatted strings. The complexity of document is then reduced. With the formatted document, more complex concept can then be extracted much more easily. In the following we will introduce the hierarchical description and the corresponding algorithm in detail.

### 8.2 Hierarchical Concept Extracting

Having the observation described above, we developed the new approach: hierarchical concept description and extraction. Base on token-based rule (see Section 7.2.2), we define a hierarchical concept as following: each concept (hierarchical) consists of four part: a relevance number indicates the relevance degree, a Core define the underlying value, a Precore defines the value direct before the concept and a Postcore defines the value direct after
the concept. The relevance and Core are must parts, while Pre- and Postcore are optional. The Core, Pre- and Postcore consist of a sequence of tokens, skip operators and lower level hierarchical concept again. The definition of token, Skip-Operator and special characters are the same as the defined for Token-based rule (see Section 7.2.2). The formal definition of the hierarchical concept is made in Figure 37.

\[
\begin{align*}
\text{Concept} & := \text{Relevance Precore} \{ \text{Core} \} \text{Postcore} \\
\text{Precore} & := (\text{Token} | \text{Skipoperator} | \text{Search Profile})? \\
\text{Core} & := (\text{Token} | \text{Skipoperator} | \text{Search Profile}) \\
\text{Postcore} & := (\text{Token} | \text{Skipoperator} | \text{Search Profile}) \\
\text{Token} & := \text{Digit} | \text{String} | \text{Special Character} \\
\text{Skipoperator} & := \{ \text{Number of skip tokens} \} \\
\text{Digit} & := \{0-9\}+ \\
\text{String} & := \{a-z,A-Z\}+ \\
\text{Special Character} & := \text{ASCII Characters} \setminus (\text{Digit, String}) \\
\text{SearchProfile} & := \{\text{Full Text Rules, Similarity Rules, Token-based rules, keywords rules, Concept}\}
\end{align*}
\]

\textbf{Figure 37: Formal Definition of Hierarchical Concept}

To explain the hierarchical concept rule we give an simple example. Having a sentence as \textit{The “hike to the top of Mt Finkol takes 8 to 10 hours”}. We can define concepts stepwise:

\begin{align*}
\text{Step 1: DIS\_VALUE: regular express “[0-.9]+( )+to( )+[0-9]+”} \\
\text{Step 1: DIS\_UNIT: full text rule with thesaurus, which contain the units such as “hours”} \\
\text{Step 2: DISTANCE: hierarchical concept <DIS\_VALUE><DIS\_UNIT>} \\
\text{Step 2: Hiking\_Place: full text rule with thesaurus, which contain the units such as “Mt Finkol”} \\
\text{Step 2: Hiking\_Activite: full text rule with thesaurus, which contain the units such as “Hiking”} \\
\end{align*}

The extraction of a hierarchical concept is processed also stepwise. In the extraction phase a defined profile is loaded. The extraction function must be called recursively if the rule for a complex concept contains rules for other more basic concepts. The basic idea is extracting and replacing. If a hierarchical concept is applied, system checks at first if the concept contains some lower level concepts. If so, system loads the lower level concept and extracts it. Because the lower concept can also be a hierarchical concept, this step can be processed more times until the primitive concepts are reached. The results of extracting lower level concept are then replaced with unique
Information Extraction

token string in document. After extracting and replacing of lower level concept, system replaces also the lower level concept names in hierarchical concept with the corresponding unique token strings. Finally, the hierarchical concept can be applied because there are no more lower concepts in the rule. The algorithm for extraction is described as follows:

<table>
<thead>
<tr>
<th>Extracting_Rule (ConceptName)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get each MatchPattern from search profile of ConceptName</td>
</tr>
<tr>
<td>While MatchPattern contains references to other more basic concepts:</td>
</tr>
<tr>
<td>Get the ContainedConceptName</td>
</tr>
<tr>
<td>Extracting_Rule (ContainedConceptName)</td>
</tr>
<tr>
<td>Replace all result with $ContainedConceptName$ in document</td>
</tr>
<tr>
<td>Make same replacement in the MatchPattern</td>
</tr>
<tr>
<td>End While</td>
</tr>
<tr>
<td>Do Extraction with processed MatchPattern in the processed document</td>
</tr>
<tr>
<td>Align positions and length of results to get the right location in original document</td>
</tr>
<tr>
<td>End Extracting_Rule</td>
</tr>
</tbody>
</table>

8.3 Learning Algorithm

Typically, complex content can be decomposed into strong patterns involving more basic concepts. This is the basic assumption underlying the idea of the hierarchical learning methodology. Firstly, rules are learnt for simple/basic concepts (such as place name, date, distance value, etc.). Such basic concepts are then replaced in the initial document with uniform token strings. This normalized text can then be used to derive more complex rules on the basis of trained simpler rules and thus to extract more complex concepts. In principle, this process could be performed successively until highly abstract concepts are extracted or until the whole document content has been captured. In practice, the process of deriving complex rules from more basic rules may cause a gradual amplification of noise that limits the degree of nesting that can be achieved in practice. However, in general, hierarchical learning allows more complex concepts to be extracted than would be possible using flat rules that are derived solely from the original source text. The algorithm for hierarchical learning is described in the following section.

This section introduces algorithms to generate hierarchical concept automatically and to extract contents with hierarchical concepts. Examples are illustrated in order to explain how hierarchical concept functions.

Hierarchical learning extends the basic algorithm by using the NE-Agent dependencies to govern the order in which NE-Agents is trained. Once each
lower level NE-Agent has been trained it is applied to the source text and each occurrence found is replaced by a normalised representation of the following form: $\text{ConceptName}$. Note, this symbolic normalisation is superimposed as a virtual view on top of the original document text, thus the original surface text is always accessible.

The input for learning is as over a set of examples defined by start positions and lengths. The learning step of each feature is given by user. For each iteration the learning algorithm contains three steps: At first the system extracts all information specified by the learned rules from the sample document. The results are defined by start position and length. At the second step the results are replaced by some unique tokens. All position and length information are updated to adjust on the current processed document. At the last step the system uses token-based Rule Extractor to learn new rules based on processed document.

For each step
- Get example set at this step
- Extract all information defined by learned rules
- Replace the results with unique token string of form $\text{ConceptName}$
- Update the position and length in example set
- Train the example based on processed document
- IF trained rule contain unique token string
  - Replace token string as trained rule name
- End if
- Give the rule a name and set it in the trained rules set

END For

8.4 Examples

In following we give an example to illustrate how rules are generated hierarchically based on a semi formal text of English Football Results archived from the 1996/1997 season. To be extracted are teams and match results. An example of the original document is:

```
NOTTINGHAM FOREST 0-1 EVERTON 19,892 Match Report
  Full-time score and scorers for Sunday, 10. 27 1996
LIVERPOOL 2-1 DERBY COUNTY 39,515 Match Report
  Full-time scores and scorers for Saturday, 10 26 1996
ARSENAL 0-3 LEEDS UNITED 38,076 Match Report
CHELSEA 3-1 TOTTENHAM HOTSPUR 28,373 Match Report…
```
The goal is to extract from each match, the teams playing and the result (more fine grained analysis such as differentiating home and away teams and number of goals each scored could obviously be achieved in a similar manner, but things are kept simple here for illustration purposes). Firstly, at level 1 the team names are learnt (this is most easily learnt using the keyword extraction agent, i.e., as a set of synonyms):

| Step 1: Train all teams, Trained Rules Set: empty |
| Step 2: Train the result, Trained Rules Set: TEAM |
| Replace all instances of TEAM in the document with token string "TEAM". |

Thus, after the concept of TEAM has been learnt and replaced in the original document, the document content appears as follows:

<table>
<thead>
<tr>
<th>TEAM</th>
<th>0-1</th>
<th>TEAM</th>
<th>19,892 Match Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team</td>
<td>Full-time score and scorers for Sunday, 10. 27 1996</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEAM</td>
<td>2-1</td>
<td>TEAM</td>
<td>39,515 Match Report</td>
</tr>
<tr>
<td>TEAM</td>
<td>0-3</td>
<td>TEAM</td>
<td>38,076 Match Report</td>
</tr>
<tr>
<td>TEAM</td>
<td>3-1</td>
<td>TEAM</td>
<td>28,373 Match Report</td>
</tr>
</tbody>
</table>

Thus in this case, the concept of a RESULT is dependent on the concept of a TEAM. Thus, training RESULT based on the above normalised document yields the rule, "TEAM {c $TEAMS$ $RESULT$ $TEAM$ c}". Note, that pattern "{c $TEAM$ - $TEAM$ $RESULT$ $TEAM$}" alone would possible have been sufficient, but using "TEAM" as per and post delimiter is more reliable. In a similar way to the concept TEAM, each instance of this rule is replaced in the document with the symbol $RESULT$. This yields the following normalised document:

<table>
<thead>
<tr>
<th>TEAM</th>
<th>$RESULT$</th>
<th>TEAM</th>
<th>19,892 Match Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team</td>
<td>Full-time score and scorers for Sunday, 10. 27 1996</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEAM</td>
<td>$RESULT$</td>
<td>TEAM</td>
<td>39,515 Match Report</td>
</tr>
<tr>
<td>TEAM</td>
<td>$RESULT$</td>
<td>TEAM</td>
<td>38,076 Match Report</td>
</tr>
<tr>
<td>TEAM</td>
<td>$RESULT$</td>
<td>TEAM</td>
<td>28,373 Match Report</td>
</tr>
</tbody>
</table>

Finally, the concept of a MATCH can be learnt based on this normalised document. Training yields the rule for MATCH as "{c $TEAMS$ $RESULT$ $TEAM$ c}". Instances of this rule can be replaced with the symbol $MATCH$ thus allowing yet more complex concepts to be learnt while avoiding the proliferation of surface level variations. Thus the above worked example illustrates the fundamental principles of the hierarchical learning. While this result is manually configured, the results of applying the automated hierarchical learning to this data set generated a very similar set of rules to those illustrated above (see Section 4.1)
8.5 Evaluation and discussion

As a preliminary trial of the hierarchical learning system, it was applied to three document collections of varying informality: the football data (see above), a collection of rental ads (Soderland 1999) and travel descriptions from Lonelyplanet WorldGuide. The Rental Ads domain was collected from the Seattle Times online-classified ads. We downloaded the test set from RISE Repository (http://www.isi.edu/~muslea/RISE/repository.html). The Lonelyplanet WorldGuide is collected from the Web sites in Lonelyplanet.com (http://www.Lonelyplanet.com)

8.5.1 Football result

Football results data archive contains actual football scores and related information of Premiership in England for the 1996/97 season. The format of this domain is described already in Section 8.4. To be extracted are name of home team, name of away team, result and total match sentence. We used a training set with 40 examples and a test set with 400 examples. We trained concepts with three steps: at first step we trained all team names as the concept TEAM, at the second step, concept RESULT is trained with training result. Finally, the total MATCH is trained at step three. The training results are a set of rules similar to those reported in the proceeding worked example. Evaluation of the system yielded results of 99% precision and 98.6% recall. The reason why the system does not reached 100% precision and recall is that there are misspelling in the original documents. These near-perfect results are not unexpected given the highly structured nature of the data – hence this experiment demonstrated that the hierarchical learning passes the minimal performance criteria.

8.5.2 Rental Ads

The motivation behind the Rental Ads domain was to enable a direct comparison to experiments previously carried out on the WHISK (Soderland 1999) system (see also the previous evaluation in Section 7.4.2). The template for information to be extracted consists of three content elements: information about neighbourhoods, about price and about the number of bedrooms. Below is given a sample of the description with an indication of what information should be extracted. The result can be output as:

Neighbourhood: Ballard  NumberOfBeds: 2  Prise: $1195
Neighbourhood: Belltown  NumberOfBeds: 1  Prise: $775 - $895
Two experiments are made with this data set. In the first experiment, the hierarchical concept learning is disabled – i.e. learning is performed based on the base text. In the second experiments hierarchical learning is enabled. Indeed, a new concept is introduced, BEDROOM, which is first trained to represent the synonyms of bedrooms (such as bd, bds, and bdrs ect.). With this concept in place the rules were trained to extract the three desired content elements.

As for WHISK (Soderland 1999) a training set of 400 examples was used and a test set of a further 400 examples. In the first experiment recall begins at 88.4% from 50 examples and climbs to 95.0% when 400 examples. Precision starts at 87.0% from 50 examples and reaches to 97.7% when 400 examples. In the second experiment the precision ranges slightly between 96.7% and 97%, while the recall becomes from 89.8% at 50 examples to 96.0% when 400 examples. The results (after NE-Agent) are shown in comparison with WHISK as following:

<table>
<thead>
<tr>
<th></th>
<th>Example</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHISK</td>
<td>50</td>
<td>94%</td>
<td>73%</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>98%</td>
<td>94%</td>
</tr>
<tr>
<td>Experiment without hierarchical training</td>
<td>50</td>
<td>73.0%</td>
<td>91.4%</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>90.2%</td>
<td>97.0%</td>
</tr>
<tr>
<td>Experiment with hierarchical training</td>
<td>50</td>
<td>94.2%</td>
<td>91.3%</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>95.0%</td>
<td>97.0%</td>
</tr>
</tbody>
</table>

**Table 10. Result of Rental Ads (After NE-Agent)**

At the first experiment, there is good recall and poor precision after NE-Agent processing. At the second experiment, the precision already reaches to 94% when 50 examples after NE-Agent. Fig. 5 shows the total learning curves for the Rental Ad domain.

---

6 The result of precision and recall in this table is not the final result. The final result after TE-Agent is not shown because it is not relevant to hierarchical learning approach.
Figure 38: Learning Curve for Rental Ads (After NE-Agent processing)

For this straightforward data set, what is demonstrated is that hierarchical learning and flat learning have little difference in terms of the eventual accuracy (precision and recall) that can be attained, but that the hierarchical learning algorithm reaches this high level of accuracy very much more rapidly than the flat learning approach.

8.5.3 Lonelyplanet

Lonelyplanet WorldGuide describes exiting travel information in the world. We gathered web pages about attractive activities of the tourist areas, which describe the destination of the area, types of activities (such as fishing, swimming, hiking etc.). As a first experiment on processing this less well-structured data, information concerning hiking, e.g. the distance of a specific hike, was extracted. The descriptions of such information are highly varied: some are absolute distance values (e.g. "The most popular hike on the continent is the 33km (20mi) Inca Trail, west of Cuzco"), some are described as walking hours (such as "The hike to the top of Mt Finkol takes 8 to 10 hours"), some are only fuzzy values (e.g. "There are dozens of day hikes and multi-day hikes throughout the island."). This information is embedded in documents with a lot of other distance information (distance between towns, height of mountains, etc.). To be extracted are value of distance, unit of distance and also the short text description about hiking distance.
The training set of this experiment is 50 examples, while the testing set contains 260 Lonely Planet WorldGuide descriptions. Again, two experiments were performed. In the first hierarchical learning was disabled. The input was a set of examples with position information; the output was a set of training rules about the contents.

In the second experiment hierarchical learning was performed in four steps:

- **Step 1**, the distance unit (such as km, ft etc.) and keywords about hiking (such as *Hike*, *Hiking*, *Hikers* etc.);
- **Step 2**, the distance value (such as 10, 3000 etc.)
- **Step 3**, the combined distance information containing both distance value and distance unit (e.g. 4km, 13 miles etc.)
- **Step 4**, rules to extract the total sentence about hiking distance are trained. The following tables show evaluation result of two experiments.

<table>
<thead>
<tr>
<th>Concept</th>
<th>Level</th>
<th>P</th>
<th>R</th>
<th>Example of Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>HikingTerm (HIKING)</td>
<td>1</td>
<td>0.252</td>
<td>1.00</td>
<td>1.00 {c hiking c} 1.00 {c hikers c}</td>
</tr>
<tr>
<td>DistanceUnit (DISU)</td>
<td>1</td>
<td>0.139</td>
<td>1.00</td>
<td>1.00 {c miles c} 1.00 &lt;int&gt; {c km c}</td>
</tr>
<tr>
<td>DistanceValue (DSV)</td>
<td>1</td>
<td>0.321</td>
<td>1.00</td>
<td>1.00 trail [2] {c &lt;int&gt; c} 1.00 {c &lt;int&gt; c} [1]</td>
</tr>
<tr>
<td>Distance (DISTANCE)</td>
<td>1</td>
<td>0.352</td>
<td>1.00</td>
<td>1.00 {c &lt;int&gt; miles c} 1.00 {c &lt;int&gt; - [1] c}</td>
</tr>
<tr>
<td>HikingDistanceInfo</td>
<td>1</td>
<td>0.150</td>
<td>0.321</td>
<td>0.80 {c the [5] &lt;int&gt; [1] c}</td>
</tr>
</tbody>
</table>

**Table 11. Result of Lonely Planet without Hierarchical Learning**

<table>
<thead>
<tr>
<th>Concept</th>
<th>Level</th>
<th>P</th>
<th>R</th>
<th>Example of Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>HikingTerm (HIKING)</td>
<td>1</td>
<td>0.252</td>
<td>1.00</td>
<td>1.00 {c hiking c} 1.00 {c hikers c}</td>
</tr>
<tr>
<td>DistanceUnit (DISU)</td>
<td>1</td>
<td>0.139</td>
<td>1.00</td>
<td>1.00 {c miles c} 1.00 &lt;int&gt; {c km c}</td>
</tr>
<tr>
<td>DistanceValue (DSV)</td>
<td>2</td>
<td>0.365</td>
<td>1.00</td>
<td>1.00 {c &lt;int&gt; c} - 1.00 {c &lt;int&gt; c} &lt;DISU&gt;</td>
</tr>
<tr>
<td>Distance (DISTANCE)</td>
<td>3</td>
<td>0.459</td>
<td>1.00</td>
<td>1.00 {c &lt;DISV&gt; - &lt;DISU&gt; c} 1.00 {c &lt;DISV&gt; &lt;DISU&gt; c}</td>
</tr>
<tr>
<td>HikingDistanceInfo</td>
<td>4</td>
<td>1.000</td>
<td>0.92</td>
<td>1.00 to [10] {c HIKING} [10] &lt;DISTANCE&gt; c 0.40 {c HIKING} [5] &lt;DISTANCE&gt; c 0.40 {c HIKING} [10] &lt;DISTANCE&gt; c</td>
</tr>
</tbody>
</table>

**Table 12. Result of Lonely Planet with Hierarchical Learning**

Clear benefits in terms of increased Recall/Precision are demonstrated in these results. Especially for the complicated concept such as
“HikingDistanceInfo”, the normal training algorithm is not more able to generate reasonable extracting rules.

8.5.4 Discussion

In this section, the key points raised by the above experiments will be discussed. The evaluation clearly demonstrates that hierarchical concept learning generally improves overall precision and, to a lesser extend, overall recall (e.g. HikingDistanceInfo). Furthermore, the Rental Ads experiment clearly shows an accelerated rate of hierarchical learning with respect to flat learning.

At first we analyze the experiment football. In this experiment, system reaches very good precision and recall. If no hierarchical concept is used, training could be difficult because of the variation of element football team. We can not get the comparable result even when we set more than 200 examples manually, while with help of hierarchical concept learning, extracting performance is already satisfying when 40 examples. That means, hierarchical learning does not only speed up leaning processing, but also improves learning quality. Because either home team or away team are extracted with concept TEAM, NE-agent is not able to distinguish these two template elements. This task is overtaken by TE-agent, which analyses the spatial relationship under template elements (see Section 9 for more detail).

For the Rental Ads domain, both the performance of both the hierarchical learning approach and the WHISK benchmark are relatively flat – i.e. reach high levels of precision and recall with relatively few training examples. In the case of WHISK background knowledge is introduced before extracting, as a so called semantic class. For instance, in the Rental Ad domain the semantic class Bedroom has been defined as the disjunction:

$$\text{Bedroom} = \{ \text{brs} | \text{br} | \text{bds} | \text{bdrm} | \text{bd} | \text{bedrooms} | \text{bedroom} | \text{bed} \}$$

The introduction of this background knowledge trivialises the extraction task. The high performance of the hierarchical learning can be explained in the same way: the pre-trained concept BEDROOM used as a first step makes the other content elements much easier to extract. The difference is that the BEDROOM concept is represented as a set of examples rather than being manually encoded. Thus the conclusion is that in order to enable high accuracy information extraction by hierarchical learning, care should generally be taken to select and define (through example) basic concepts that will simplify the learning task through the normalisation of the raw text.

Another advantage of hierarchical learning is that we can get satisfactory performance (precision and recall) with less training examples than traditional
learning algorithm. By replacing some contents with formatted token strings the machine learning system can have standardised information as training input. These standardised inputs can reduce the variation of contents and perform a better and quicker learning result.

The above discussion shows that higher levels of accuracy can be attained more rapidly through hierarchical learning because of the normalising effect of the basic concepts used to define more complex concepts (e.g. `HikingDistanceInfo` defined in terms of `DISTANCE`). This also introduces another methodological advantage, namely that of reuse. The same basic concepts can be used to define related complex concepts. Hence, `DISTANCE` can be reused to define `DistanceBetweenCities`, etc. In this way, the knowledge engineering required for large-scale information extraction projects can be minimised.

The hierarchical learning approach also provides a generic framework by which different information extraction techniques can be combined – the basic assumption being that any information extraction technique carries out a normalisation operation on the raw text. Taking this idea one step further, the use of NLP to carry out, for example POS tagging or stemming as a pre-processing operation for hierarchical concept learning would be a next extension. The investigation of how different machine learning techniques and/or NLP techniques can be most favourably combined in this manner is left to future work.

### 8.6 Related Work

As noted, replacing formatted strings in original documents are used already in WHISK (Soderland 1998). Soderland defined semantic class to replace e.g. the `"br"` string by the semantic class “Bedroom”, which is defined as

\[
\text{Bedroom ::= ( br; brs; bdrm; bedrooms; bedroom )}
\]

That is, the semantic class “Bedroom” is a placeholder for any of the abbreviations above. Actually, the semantic class in WHISK is very similar as concept in CapturePlus. The difference between WHISK and Capture is that WHISK’s semantic class is defined manually, while CapturePlus generates concepts from examples automatically. Another advantage in CapturePlus is the hierarchical approach. WHISK replaces strings in original documents with semantic class only in the lowest level. CapturePlus does it in iterative way (i.e., not just synonym replacement).

Some systems support also stepwise extracting similar to the hierarchical approach. FAUST (Appelt 1993) used cascading finite state automata to perform stepwise extracting. FASTUS is a system for extracting information
from natural language text for entry into a database and for other applications. It works essentially as a cascaded, non-deterministic finite-state automaton. There are five stages in the operation of FASTUS. In Stage 1, names and other fixed form expressions are recognised. In Stage 2, basic noun groups, verb groups, and prepositions and some other particles are recognised. In Stage 3, certain complex noun groups and verb groups are constructed. Patterns for events of interest are identified in Stage 4 and corresponding “event structures” are built. In Stage 5, distinct event structures that describe the same event are identified and merged, and these are used in generating database entries. This decomposition of language processing enables the system to do exactly the right amount of domain-independent syntax, so that domain-dependent semantic and pragmatic processing can be applied to the right larger-scale structures. However, FAUST is not able to abstract concept in higher levels.

Another system STALKER (Muslea 1999) tries to analyse hierarchy in online-documents. For instance, the information on a page often exhibits some hierarchical structure, furthermore, semi-structured information is often presented in the form of lists of tuples, with explicit separators used to distinguish the different elements. The embedded catalogue (EC) can describe the structure of a wide-range of semi-structured documents. The EC description of a page is a tree-like structure in which the leaves are the item of interest for user. The internal nodes of the EC tree represents lists of k-tuples, where each item in the k-tuple can be either a leaf or another list (hierarchical representation). However, this hierarchical representation is merely a generic template definition, which has nothing to do with hierarchical concept description. The extraction rules behind each corresponding node is simple syntactic rules, which are interpreted as finite state automata. Actually, the hierarchy in STALKER is defined in TE-level but not in NE-level.