From the Use of Words in Natural Language to an Ontological Commonsense Structure

Ruth Janning

FernUniversität in Hagen, Lehrgebiet Wissensbasierte Systeme, Universitätstraß 1, 58097 Hagen, Germany
ruth.janning@fernuni-hagen.de

Abstract. To simulate human reasoning, artificial intelligent systems need background knowledge. This knowledge can be represented by an ontology. For specific domains domain ontologies [1] are used. But human inferences beyond the boundaries given by specific domains can only be imitated by means of commonsense knowledge. However, the generation of commonsense ontologies is a very difficult challenge [1]. Existing commonsense ontologies (e.g. Cyc [2]) were constructed over a lengthy period of time. Hence there is need for other methods for gaining commonsense ontologies in shorter time with less effort. A very interesting proposal is to expose the structure of commonsense knowledge (instead of constructing it with immense effort) by analyzing the use of words in natural language [4].

Keywords: Commonsense, Knowledge Engineering, Natural Language, Ontology, Reasoning

1 About Ontologies and Commonsense Ontologies

A human being learns from childhood throughout his whole life and memorizes his experiences as background knowledge. This stored knowledge is useful for acting and inferring in everyday situations, e.g. to know that a dog, which toyed in water, is wet and does not smell very well. A Computer does not possess such background knowledge by itself. Hence, a representation of background knowledge for artificial intelligent systems is needed. For this purpose commonsense ontologies can be employed.

The expression “ontology” has philosophical origins and was used to describe and classify the existence of things in the world [1]. In computer science ontologies structure and formally represent knowledge. T. R. Gruber [5] defines an ontology in computer science as “a formal, explicit specification of a shared conceptualization”. “Formal” means that an ontology must be a formal representation, that could be processed by machines. The word “shared” implies the common view of the considered domain. “Conceptualization” corresponds to

---

1 Knowledge was fed over 20 years by human knowledge engineers into the knowledge base of Cyc [3] until the system was able to learn by itself.
the definition of the concepts and the relations between them. This definition implies so called domain ontologies [1]. If not one special domain is regarded but our whole world we live in, commonsense ontologies are gained.

2 Four Steps Toward an Ontological Structure

The construction of commonsense ontologies with existing methods causes an immense effort. Hence there is need for new methods to gain these ontological structures. Saba [4] suggests a method to expose - and not newly develop - the structure of commonsense knowledge by analyzing the use of words in natural language. Because there exists a relationship between natural language understanding and knowledge representation in conjunction with reasoning. Due to the ‘understanding as reasoning’ paradigm [4], natural language understanding is a commonsense reasoning process at the pragmatic level.

The analysis of words is executed on a very simple base - the view of a child. Because asking a child, if it makes sense to say “a dog barks” or “a book barks”, one gets an explicit positive or negative answer. Accordingly quantitative or philosophic assessments, which an adult would maybe have in mind, are not regarded (e.g. there are dogs that do not bark or such dogs, who bark louder or lower). Now it is a binary decision if it in general makes sense to say “a dog barks” or not.

Introducing a predicate $App(p, c)$, Saba [4] describes a process which guides within four steps toward an ontological structure. In the expression $App(p, c)$ $p$ is a property (adjective) or action (verb) and $c$ a concept (substantive). $App(p, c)$ gets the value true, if it makes sense to speak of the property or action $p$ of $c$. So the following steps guide toward an ontological structure:

1. Assume a set of concepts (substantives) $C = \{c_1, ..., c_n\}$ and a set of properties (adjectives) or actions (verbs) $P = \{p_1, ..., p_m\}$ to be already known.
2. Furthermore a predicate $App(p, c), c \in C$ and $p \in P$, is given. $App(p, c)$ becomes true if the property or action $p$ is reasonably applicable to objects of type $c$ (i.e. if it makes sense to speak of the property or action $p$ of $c$).
3. For every property or action $p \in P$ a set $C_p = \{c | App(p, c)\}$, which includes all concepts $c$ for which $App(p, c)$ is true, is generated.
4. As a result, the wanted hierarchy is gained through an analysis of the subset relationship between the sets generated in step 3.

3 Data mining, a Table and an Algorithm

This four step process proposed by Saba [4] is an interesting beginning and should be further developed. Hence now suggestions for realizing the four steps will follow.

\[^{2}\text{If to the question, whether this combination makes sense, a child would answer “yes”.}\]
3.1 Step 1 and 2: Data mining

To choose the substantives and adjectives or verbs for the sets $C$ and $P$ and to generate a table, which represents the combinations of substantives and adjectives or verbs, is a difficult challenge. First one has to find enough and above all enough different (in their meaning) words. Because every concept of the resulting hierarchy must possess a unique characteristic to be different from other concepts. This can be reached by means of meaningful adjectives or verbs.

The proposal here is to instruct a computer to analyze texts (e.g. from the numerous sources of the Internet like Wikipedia [6]). The computer could search for adjectives, verbs and substantives by using computerized dictionaries. Finally the computer should explore the combinations of adjectives or verbs and substantives with data mining techniques by checking how often some combinations occur (checking the support of associations [7]). Often occurring combinations (with higher support than other combinations\(^3\)) could get the symbol “+” in the corresponding table cell (that means $\text{App}(p, c)$ is true for this combination). Combinations with a support below the determined threshold receive a “-”.

3.2 Step 3: Table

The analysis of the combinations of substantives and adjectives or verbs via $\text{App}(p, c)$ in step 2 could be represented by the table mentioned above. The substantives are inserted in this table in the first column and the adjectives or verbs in the first row\(^4\). An inner table cell receives the symbol “+”, if $\text{App}(p, c)$ is true for the corresponding combination of $p$ and $c$. Otherwise the table cell gets a “-” (see example 1 and table 1).

Example 1.

\begin{align*}
C &= \{\text{bird, book, dog, machine, man, woman}\} \\
P &= \{\text{bark, big, defective, fly, live, pregnant, read}\}
\end{align*}

Table 1. Table for example 1.

<table>
<thead>
<tr>
<th></th>
<th>bark</th>
<th>big</th>
<th>defective</th>
<th>fly</th>
<th>live</th>
<th>pregnant</th>
<th>read</th>
</tr>
</thead>
<tbody>
<tr>
<td>bird</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>book</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>dog</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>machine</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>man</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>woman</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

Consequently the sets $C_p$ from step 3 can be obtained by picking out the table column corresponding to $p$ (see table 2).

\(^3\) One has to define a reasonable threshold.
\(^4\) Both in alphabetical order.
Table 2. Sets $C_p$ and their complementary set pertaining to the father set (see section 3.3).

<table>
<thead>
<tr>
<th>Set</th>
<th>Complementary Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{dog}$</td>
<td>{bird, book, dog, machine, man, woman}</td>
</tr>
<tr>
<td>$C_{live}$</td>
<td>{bird, dog, man, woman}</td>
</tr>
<tr>
<td>$C_{read}$</td>
<td>{man, woman}</td>
</tr>
<tr>
<td>$C_{defective}$</td>
<td>{machine}</td>
</tr>
<tr>
<td>$C_{pregnant}$</td>
<td>{woman}</td>
</tr>
<tr>
<td>$C_{bark}$</td>
<td>{dog}</td>
</tr>
<tr>
<td>$C_{fly}$</td>
<td>{bird}</td>
</tr>
</tbody>
</table>

3.3 Step 4: Algorithm

Gaining the hierarchy of substantive sets can be realized by the following algorithm in pseudo code.

**Algorithm Hierarchy Generation**

**Method:** generateHierarchy()

**Output:** Concept hierarchy.

**Algorithm:**

1. Initialization of the root node of the concept hierarchy with the set, which contains all concepts.
2. Call of the recursive method getSubSets(<set>) with the set of the root node as its argument: getSubSets(root node set)
3. return concept hierarchy

**Method:** getSubSets(<set>)

**Given:** Discovered sets $C_p$ of concepts (see above step 3).

**Input:** Set, of which the largest subset and its complementary set, pertaining to the set, are to determine and to insert as left and right son.

**Algorithm:**

1. Find all real subsets of <set>.
2. Determine the largest set of these subsets (if there are more than one, choose those, of which the related property or action is alphabetically smaller than all the others).
3. LeftSonNode = determined largest set.
4. RightSonNode = <set> \ LeftSonNode.
5. if LeftSonNode is not equal to the empty set then
   a. Insert in the hierarchy LeftSonNode as left son of the node, which is represented by <set>.
   b. Determine the subsets of the left son: getSubSets(LeftSonNode).
c. Insert in the hierarchy RightSonNode as right son of the node, which is represented by <set>.

d. Determine the subsets of the right son:
getSubSets(RightSonNode).

This algorithm searches the largest subset to a node of the hierarchy (respectively to the corresponding set of substantives) and inserts these as his left son in the hierarchy. The reason for choosing the largest subset is that the sets, which represent nodes, should become smaller downward. That is they should become more and more particular and represent more particular concepts. The right son node is the complementary set of the left son node pertaining to the set in the father node. Because a property or action is supposed to separate the set of a node into two complementary sets, which finally represent its son nodes in the hierarchy. The result is a hierarchy like the one shown in figure 1. In such a hierarchy all son nodes and descendants are sub concepts of their father node. So the root node includes all concepts.

\[\text{Fig. 1. Generated ontological structure.}\]
**Decision Trees** The presented algorithm is very similar to the generation of decision trees [7]. Methods to get decision trees generate such trees from sets of examples. An example possesses a set of attribute-value pairs and a classification. These examples correspond to the concepts (substantives) an the attributes to the properties (adjectives) or actions (verbs). But there are only two different values for attributes ("+" or "-"") in the method proposed in this paper.

The difference between the two methods is, that the classification of concepts is not known beforehand. In addition, the presented algorithm does not end in a leaf because all examples possess the same classification but because there is no more subset for this node.

For decision trees a heuristic exists to get as small trees as possible [7]. It says that the most important attribute must be tested first. The most important attribute should cause the best differentiation of the example set. Corresponding to this, in the presented algorithm the most important attribute is the action or property, which is applicable to the most concepts. As this represents the largest subset.

**Rules** With a hierarchy gained by means of the algorithm, one can get logical rules. The hierarchy in figure 1 for instance offers the following rules.

*Example 2.*

\[ \forall c (\text{App(pregnant, } c) \Rightarrow \text{App(live, } c)) \]

For all objects, which can be *pregnant*, it makes also sense to say they *live*.

*Example 3.*

\[ \forall c (\text{App(bark, } c) \Rightarrow \neg \text{App(read, } c)) \]

For objects, which can *bark*, it does not make sense to say they *read* (In short: Dogs cannot read.).

*Example 4.*

\[ \exists c (\text{App(live, } c) \land \neg \text{App(fly, } c)) \]

There are objects, which *live* but do not *fly*. (Not every living being is able to fly.)

**Metaphorical Meanings** If substantives and adjectives or verbs are analyzed by \textit{App}(p, c), it is important to use the original meaning of a word and not a metaphorical meaning (see example 5 and 6).

*Example 5.* On the one hand the word *heavy* has the original meaning “with weight” and on the other hand it means something like “oppressively”.

*Example 6.* The word *ball* represents originally an “object with a spherical shape to play with” but also a “formal dance event”.


Several meanings of a word must absolutely be distinguished. The different meanings must appear at different places in the hierarchy. This is not avoidable, if one wants to get an unambiguous and reasonable ontological structure.

But this causes a problem for the text analyzed by a computer. In normal texts words are used in their original meaning as well as in metaphorical meanings. But possibly the usage of data mining techniques is already a solution. These methods give an “+” only to combinations of substantives and verbs or adjectives, which reach a high support. So if metaphorical meanings compared to the original meaning occur scarce, they will be ignored.

4 Conclusions

The proposed method is a good beginning, but there are still problems. A very large number of substantives and adjectives or verbs need to be cumulated to get a useable ontological structure. And to gain a full commonsense ontology, these words must be sufficiently different from each other.

In addition to this, the use of the method is not always unmistakable, because it is difficult to say in which case an adjective or verb is really reasonably applicable to a substantive. Further problems result from the need to separate original and metaphorical meanings of words which must appear at different places in the hierarchy.

However, the method is a promising start to construct commonsense ontologies with less effort (and automatically by a computer) than currently existing methods.

References