AGENT-SUPPORTED INFORMATION RETRIEVAL FOR NATURAL LANGUAGE

Ute Loerch & Hans W. Guesgen
Computer Science Department, University of Auckland
Private Bag 92019, Auckland, New Zealand
E-mail: ute, hans@cs.auckland.ac.nz

ABSTRACT

This paper addresses issues in using modeling techniques for a multi-agent system, which adapts and extends existing object-orientated (OO) representation techniques for parsing natural language. After concentrating on the main problems of natural language information retrieval, we will focus on the details of the applied agent-orientated (AO) modeling techniques and assume a passing familiarity with OO modeling and representation techniques.

In addition to the above, we discuss the utility of having a matchmaking agent which can reason over agent capabilities to recommend agents for specific tasks, where the capabilities and requirements are defined using a common service ontology. This ensures that the semantics of matching agent capabilities remains the same across the multi-agent system.

INTRODUCTION

Object-modeling techniques [3,4] describe a system by identifying the key object classes in an application domain, and specifying their behaviour and their relationships with other classes. The essential details of a system design are captured by three different types of models: an object model, a dynamic model and a functional model. By contrast, in specifying an agent system of a more specialized set of models, which operate at two distinct levels of abstraction, seems to be a lot more efficient.

Firstly, from an external viewpoint, the system is decomposed into agents, modeled as complex objects characterized by their purpose, their responsibilities, the information they require and maintain, and their external interactions. Secondly from the internal viewpoint, the elements required by the agent architecture must be modeled for each agent. In our case these are the agent’s beliefs, goals and plans to do natural language analysis.

The description of an agent system from an external viewpoint is captured in two models:

- An agent model that describes the hierarchical relationship among different abstract and concrete agent classes, and identifies the agent instances which may exist within the system, their multiplicity, and when they come into existence.
- An interaction model that describes the responsibilities of an agent class, the services it provides, associated with interactions and control relationships between agent classes.

From the internal viewpoint, each agent class is specified by three models, which describe its informational and motivational state and its potential behavior:

- A belief model describes the information about the environment and internal state that an agent may hold, and the actions it may perform.
- A goal model describes the goals that an agent may possible adopt and the events to which it can respond.
- A plan model describes the plans that an agent may possible employ to achieve its goals or respond to events it perceives. It consists of a plan set, which describes the properties and control structure of individual plans.

In this paper we will discuss how these models have been applied to the recovering of the structure of natural language input. Since sentences are not the only linguistic objects that possess internal structure, the definition of the belief, goal and plan of the agents is based on the variety of structures that have to be analyzed. Other objects like words, clauses, and phrases are also structured [5]. During parsing, the linguistic structure is recovered at each level of analysis.

Many different techniques for natural language processing have been described in the literature, ranging from AI methods like conceptual analysis [6] to grammar based techniques such as HPSG [7]. But when these techniques are scaled up and applied to a wider range of vocabulary and grammatical structures, the parsing mechanisms are placed under severe strain by the weight of multiplying ambiguities [8].
There are a number of basic issues in natural language parsing. They involve determining how many different types of grammatical constructions should be covered, how much ambiguity arises as the result of more or less grammatical coverage, and determining the amount and nature of the information required solving the resulting ambiguity.

Before presenting the framework for modeling and specifying our complex multi-agent system, we will be concentrating on a few important issues related to natural language information retrieval.

2. NATURAL LANGUAGE AMBIGUITY

Natural language parsing is the process of recovering the structure of natural language input. Sentences are not the only linguistic objects that possess internal structure. Other objects like words, clauses, and phrases are also structured. Their resulting hierarchy of levels of linguistic structure is summarized in Figure 1. During parsing the linguistic structure is recovered at each level of analysis.

<table>
<thead>
<tr>
<th>Analysis Level</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>-Morphology</td>
<td>-Structure of words</td>
</tr>
<tr>
<td>-Phrase</td>
<td>-Structure of noun phrases, adjective phrases, verb</td>
</tr>
<tr>
<td>-Clause</td>
<td>-phrases</td>
</tr>
<tr>
<td>-Sentence</td>
<td>-Structure of verb-argument combinations</td>
</tr>
<tr>
<td>-Modifier Attachment</td>
<td>-Structure of clause combinations</td>
</tr>
<tr>
<td></td>
<td>-Attachment of optional modifiers to phrases</td>
</tr>
</tbody>
</table>

Ambiguity occurs at each level of analysis, and it has the potential to multiply across levels. There are many decision points during natural language analysis that give rise to ambiguity, such as determining word-sense, subcategorization, complementation patterns, scope of quantification and negation, and more.

2.1 Ambiguity and Robust Parsing

There are a number of basic issues in natural language parsing. The following are the ones we concentrate on:

- **Grammatical Coverage**
  In constructing a parsing grammar, a larger or smaller number of grammatical constructions can be covered. For example, the grammar can be tightly restricted to define the notion of 'sentence' in terms of a small number of well-formed sentence structures, or it can be less restricted and can also admit some fragmentary and incomplete structures. The ambiguity inherent in a language model can be measured by the perplexity (or entropy) of the model.

- **Ambiguity Resolution Schemes**
  A 'high coverage' grammar with high perplexity provides more coverage, but results in higher degrees of ambiguity. This relationship holds for other knowledge sources and during analysis as well. For example, lexical knowledge sources (such as dictionaries and morphological analysis routines) that cover more forms cause higher lexical ambiguity. An ambiguity resolution scheme requires an information source to determine which choices are more likely, and thereby helps to disambiguate the sentence analysis. Experience has shown that strict syntactic constraints and semantic preference rules are difficult to scale up to wide-coverage parsing. Therefore we have taken the more flexible approach of using multi-agents, as described earlier on, that allows a more flexible handling of the ambiguity.

- **Statistical Modeling for Ambiguity Resolution**
  Ambiguity Resolution is the biggest problem for robust natural language analysis. It has proven difficult to achieve robust language analysis using traditional approaches that rely on hand-coded syntactic constraints and domain-specific semantic and pragmatic rules for any large or open-ended domain. Probabilistic models have got the advantage of robustness, since they provide a measure of generalization beyond the training data. If appropriate training data are available, these models surpass simpler models in terms of measured performance on several ambiguity resolution tasks. Unlike the knowledge-based approach, they only require the identification of potentially relevant features.

3. MODELING OF NEW WORDS

Since the agents of the parser are not trying to understand the meaning of the linguistic structure, it is possible to deal with words that have never been encountered before. The unknown word model therefore excepts a word about which there is no information in the computational lexicon.

The modeling procedure is as follows:

- A set of features and possible values for the features is determined.

- Each word/tag combination is classified according to the set of features.

One way to determine the set of features is to have an inflection feature; e.g. does the word carry one of the following inflection suffixes? The possible values for this feature are: -ed, -er, -est, -ing, -ly, -s. Figure 2 shows what the different inflectional suffixes stand for. Other features
defined are: INCLUDES_NUMBER, CAPITALIZED, INCLUDES_PERIOD, INCLUDES_COMMA, INCLUDES-HYPHEN, SENTENCE-INITIAL, ALL-UPPER-CASE, SHORT, PREFIX and SUFFIX.

### Meaning

<table>
<thead>
<tr>
<th>Meaning</th>
<th>Inflectional Suffix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comparative Adjective</td>
<td>-er</td>
</tr>
<tr>
<td>Superlative Adjective</td>
<td>-est</td>
</tr>
<tr>
<td>Adverb</td>
<td>-ly</td>
</tr>
<tr>
<td>Comparative Adverb</td>
<td>-er</td>
</tr>
<tr>
<td>Superlative Adverb</td>
<td>-est</td>
</tr>
<tr>
<td>Past Form Verb</td>
<td>-ed</td>
</tr>
<tr>
<td>Present Participle Form Verb</td>
<td>-ing</td>
</tr>
<tr>
<td>Verb</td>
<td>-ed</td>
</tr>
<tr>
<td>Past Participle Form Verb</td>
<td>-s</td>
</tr>
<tr>
<td>Third Person Singular Form Verb</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2: Inflectional Suffixes

The idea behind a feature like CAPITALIZED is to pick out words, like proper nouns, that are capitalized, while compensating for the fact that all sentence-initial are capitalized. This brief overview on how to model unknown words actually demonstrates, that every word can initially be treated as an unknown word. This has got the advantage of not only not having to rely as much on a lexical database, but also that speed of the parsing process has been increased.

#### 3.1 Measuring Residual Ambiguity: An Example

Suppose unknown words can be one of the following four: noun, verb, adjective, and adverb. What is the ambiguity of an unknown word? Suppose there is no additional information about it and a uniform distribution is used. In that case, each possible value has the same probability:

\[
p(noun) = p(verb) = p(adjective) = p(adverb) = 0.25
\]

Thus the entropy of a word is calculated as follows:

\[
H(Word) = -\sum_{i=1}^{n} p_i \log_2 p_i = 2
\]

So, without a model, the uncertainty associated with Word is 2 bits. The ambiguity of Word is as follows:

\[
AMB(Word) = 2^H(Word) = 4
\]

There are four equally likely choices. Now, what is the contribution of the model? Or: How much reduction of uncertainty, and therefore reduction in ambiguity, does the model provide? To determine this, the reduction in entropy of the result has to be calculated. Suppose that, for some word, the model returns the following probabilities:

\[
p(noun) = 0.65
\]

\[
p(verb) = 0.25
\]

\[
p(adjective) = 0.07
\]

\[
p(adverb) = 0.03
\]

The entropy for this word is as follows:

\[
H(Word) = -\sum_{i=1}^{n} p_i \log_2 p_i = 1.325
\]

The residual ambiguity returned by the model is 2.51. Since the original ambiguity of the problem is 4, using the results of the model has reduced the ambiguity by almost 1.5. In practice, the external evaluation measures concerning accuracy provide a more meaningful way to evaluate different models, and they are the focus of the experimental evaluations. When a model is evaluated it is applied many times, the average accuracy and entropy scores for the different instances are calculated at the end of the experiment, and the average ambiguity is calculated from the average entropy.

### 4. Modeling Individual Agents

As mentioned in the introduction, from the internal viewpoint, each agent class is specified by three models, which describe its informational and motivational state and its potential behavior.

#### 4.1 The Belief Model

A belief model consists of a belief set and one or more belief states. The belief set is specified by a set of object diagrams, which define the domain of the beliefs of an agent class.

##### 4.1.1 Belief Sets

A belief set is a set of predicates and functions whose arguments are terms over a set of built-in and user-defined type domains. These predicates, functions and domains are directly derived from the class definitions in the belief set diagrams, and the associations between them. Each class definition defines one or more domains, and each attribute, operation or association defines a predicate and/or function schema.

A belief set diagram is a directed, acrylic graph containing nodes denoting both abstract and concrete
belief classes. The classes defined in a belief set diagram correspond, in many cases, to real objects in the application domain, but unlike an OO object model, the definitions do not define the behaviors of these objects. This is because they are not implementations of the objects, rather, they represent an agent’s beliefs about those objects. Each belief class serves to define the type predicates that apply to the object, including actions, which have a special role in plans. Attributes, which define binary predicates, are specified in the same way. Predicates may also be defined by binary and higher order associations between classes.

The classes defined in a belief set diagram correspond, in many cases, to real objects in the application domain, but, unlike an OO object model, the definitions do not define the behaviors of these objects.

4.1.2 Belief Properties

The properties of predicates and functions that constitute an agent’s belief are indicated by keywords in the property list associated with the attributes, the operation, or association. Properties may also be associated with classes and instances, and may be represented by a property list.

4.1.2 Belief States

A belief state specifies a particular state of the agent’s belief in its initial mental state. It consists of a set of instances of predicates, which are specified, by a set of belief diagrams containing object instances and associations between them.

4.2 The Goal Model

Each agent class is associated with a particular Goal Model, consisting of a goal set and one or more goal states. The goal specifies the domain of the goals of an agent of that class, and the events to which it may respond. Goal states are sets of ground instances of elements of the goal set which may be used to initialize an agent’s initial mental state.

A goal set is, formally, a set of goals and events, which consist of modal goal and event operators. These operators are:

- **achieve** - denoting a goal achievement,
- **verify** - denoting a goal verification,
- **test** – denoting a goal determination.

Such activity is performed by finding the set of plans whose activation plan matches the goal, and executing one or more of them to determine the success or failure of the activity.

The different modalities determine how, exactly, such activities are performed. Depending on the modality, the type of predicate, and the agent’s initial beliefs about the predicate, different execution sequences may occur. A goal achievement succeeds if its predicate is believed to hold. Otherwise, if a matching plan executes successfully, the goal succeeds, else it fails. The postcondition of successful completion is that the predicate is believed to hold.

A goal verification succeeds if its predicate is believed to hold, and fails immediately if its predicate is believed not to hold. Only in the case that the predicate is unknown, the plan execution can result. If a matching plan executes successfully the goal succeeds, else it fails. The postcondition of successful completion is that the predicate is believed to hold.

A goal determination succeeds if its predicate is believed to hold or not to hold. Only in the case that the predicate is unknown the plan execution can result. If a matching plan executes successfully the goal succeeds, else it fails. The postcondition of successful completion is that the predicate is no longer unknown.

4.3 The Plan Model

A plan model consists of a set of plans, known as plan set. Individual plans are specified as plan diagrams. Plans may have attributes, but these may not be arbitrary, rather they are restricted to a set of predefined reserved attributes. Plans may have properties associated with them, which are specified by reserved attributes. Some of these influence the way in which the plan execution proceeds in response to a new goal or event.

- The priority property determines the order in which concurrently active plans are executed.
- The precedence property determines the order in which plans that respond to a new goal are successively tried.
- The noretry property specifies that the goal should not be retried if this plan fails.

In absence of any such properties, the set of plans applicable to a new goal is executed one by one in some arbitrary order until some plan succeeds. The precedence property allows the order in which this happens to be determined. The noretry property allows to ensure that if a certain plan is tried and fails, then no plans of lower precedence will be tried for that goal. Both these properties have important uses when modifying the behavior of inherited plans.

5. MODELING MULTI-AGENT SYSTEMS

5.1 The Agent Model

Having described the models that capture the state and behavior of individual agents, we now proceed to describe how our multi-agent is modeled.

An agent model has two components: an agent class model, which defines abstract and concrete agent classes...
and captures the inheritance and aggregation relationships between them, and an agent instance model, which identifies agent instances and their properties. In our system the number of agent classes and instances being quite small, they can be combined in only a single diagram.

An agent class model is a directed, acyclic graph containing nodes denoting both abstract and concrete classes.

Agent classes may have attributes, but no operations. Attributes may not be arbitrary, rather they are restricted to a set of predefined attributes. For example each class must have associated belief, goal and plan models, specified by the attributes beliefs, goals and plans. Multiple inheritance is permitted. Inheritance, as usual, denotes an is-a relationship, and aggregation a has-a relationship, but in the context of an agent model these have a special semantics. Agents inherit and may refine the belief, goal and plan models of their superclasses. It is rather the set of plans which is refined, rather than the individual plans.

5.2 Inheritance in the Agent Model

Inheritance is the fundamental relationship between classes which forms the basis of the structure of an agent class model.

Each agent is characterized by its belief, goal and plan models, which describe its internal state and behavior. Inheritance allows one agent class to be defined as an extension or restriction of another; the belief, goal and plan models of the superclass may be extended and specialized in the subclass.

5.3 Aggregation in the Agent Model

Aggregation is a secondary relationship between agents that allows their grouping together into a new class in quite a different way from (multiple) inheritance. The agents, called subagents, that form part of an aggregate are separate modules or name spaces; their belief, goal and plan models are quite independent. The aggregate class itself may also have belief, goal and plan models of its own. Subagents cannot directly affect each others beliefs, goals, and intentions.

A distinction between two different sorts of agents can be drawn:

- A logical agent is an encapsulation of state and behavior, developed during the system design process.
- A physical agent is the actualization of one or more logical agents.

Aggregation is a system-structuring construct that allows boundaries of physical agents in a multi-agent system to be set differently from those of individual logical agents.

5.4 Instantiation in the Agent Model

Instantiation, the third relationship that occurs in the agent model, is a relationship between agent class and one or more instances of that class. It is used to capture certain properties of agents, such as when they may come into existence and whether they will have a multiple instantiation.

An agent instance model is an instance diagram which defines both the static agent set - the set of agents that are instantiated at compile time - and the dynamic agent set - the set of agents that may be instantiated at run-time.

Therefore this agent modeling technique has got the ability to support physical agents that are themselves multi-agent systems. Figure 2 illustrated the two distinct ways of combining the two different layers.

6. AN AGENT-ORIENTATED DESIGN METHODOLOGY

This methodology supports the design and the specification of agent systems. Consideration issues such as the creation and duration of roles and their interactions determine the relationships between agent and classes.

Therefore two models have been developed that can be described in a few major steps.

- **The External Model**
  1. Identification of the application domain
  2. For each role, identify its associated responsibilities
  3. For each service, identify its associated responsibilities, and the services provided to fulfill those responsibilities
  4. Refinement of the agent hierarchy

- **The Internal Model**
  1. Analyze the means of achieving the goals.
  2. Building of the beliefs of the system.
Unlike object-orientated methodologies, the primary emphasis of the methodology we have used is on roles, responsibilities, services, and goals.

7. CONCLUSION

The primary contribution of this paper has been to provide the elements of a framework for modelling and specifying a complex multi-agent system for natural language analysis. We have given a semantics for inheritance, aggregation and instantiation relationships amongst agent classes and instances which provides a flexible mechanism for enforcing modularity of state and behavior within agents, and for sharing them between agents. Related beliefs, goals, and plans may be encapsulated in separate classes, which may then be grouped together. This paper basically presents the individual models that we have created for the different types of natural language constructions, by using an agent for each construction. By using the models described above, were every agents plan includes a residual ambiguity and the goal can only be reached, if a certain accuracy has occurred; we believe to have found a very good way to filter out words and so-called Information Entities (IE) from natural language. Therefore the parser is also able to handle the problem of new and unknown words.

REFERENCES


