Knowledge Based Conversational Agents and Virtual Storytelling

Paul Tarau[1] and Elizabeth Figa[2]

[1] Department of Computer Science
University of North Texas
P.O. Box 311366
Denton, Texas 76203
http://www.cs.unt.edu/~tarau
tarau@cs.unt.edu
[2] School of Library and Information Sciences
University of North Texas
P.O. Box 311068
Denton, Texas 76203
http://www.unt.edu/slis/people/faculty/figa.htm
efiga@lis.admin.unt.edu

Abstract. We describe an architecture for building speech-enabled conversational agents, deployed as self-contained Web services, with ability to provide inference processing on very large knowledge bases and its application to voice enabled chatbots in a virtual storytelling environment. The architecture integrates Prolog based natural language pattern matching components and story specific information extraction from RDF/XML files. Our Web interface is dynamically generated by server side agents supporting multi-modal interface components (speech and animation). Prolog refactorings of the WordNet lexical knowledge base, FrameNet and the Open Mind common sense knowledge repository are combined with internet meta-search to provide high-quality knowledge sources to our conversational agents. An example of conversational agent with speech capabilities is deployed on the Web at http://logic.csci.unt.edu:8080/wordnet_agent/frame.html.
Keywords: *conversational agents, virtual storytelling, logic programming, natural language and speech processing, WordNet, FrameNet and Open Mind based knowledge processing*

1 Introduction

Our interest in chat agents has emerged from the development of interactive Web based storytelling programs. Given the nature of storytelling performances is ephemeral and not replicable, important artifacts of world culture are being lost. In other types of work settings, corporate memory and organizational knowledge is being similarly lost or not exploited for its optimal use. Using conversational agents for storytelling has been shown to "bring to life" the collaborative computer-centered work environments necessary to sustain and make

thrive the work of distributed teams and groups or academic classes working in virtual environments. Developers approach this in a number of ways through new technologies [1], applications [2], authoring tools [3], virtual characters [4], and models for narrative construction [5].

2 Storytelling Agents

We have coded our story-telling agents as a combination of story specific Semantic Web metadata (encoded as XML/RDF [6, 7] files) and Prolog Web services integrating the WordNet [8–10] and FrameNet [11–13] lexical knowledge bases and a subset of the Open Mind [14–16] collection of common sense ontologies. For some stories, online chat transcript are used for establishing the Prolog query/answer patterns through an example driven learner implemented in Prolog, which uses WordNet based generalizations (hypernyms) to extend its coverage. The query/answer correlations extract ontology specific knowledge from the story's RDF metadata, text and related chat recordings. The agent's conversational capabilities are enhanced by matching content from WordNet, FrameNet and Open Mind. A first level in our rule hierarchy provides an essentially stateless, reactive dialog layer. Given that the reactive rules are specialized with respect to the content of a given story, this captures the most likely questions and provides them with largely predefined answers (up to a fairly flexible WordNet based semantic equivalence relation). However to provide access to the state of the interaction as well as to the content of the XML based story database, we extend the shallow pattern processing with a logic-based inference engine. The engine consists of a natural language parser, a common sense database, a lexical disambiguation module, as well as a set of transformation rules mapping surface structures to semantic skeletons in a way similar to the natural language processor described in [17]. The inference engine uses a dynamic knowledge base, which accumulates facts related to the context of the interaction. Such facts can be used for future inferences. This dynamic knowledge base works as a short-term memory similar to the one implicit in human dialogue and also provides means to disambiguate anaphoric references. On the other hand, a permanent static database obtained by scanning XML-based metadata for each story, built by a human indexer, provides more specific information, when the semantic structure of the query can be translated into predicates matching metadata tags. A fragment of such encoding follows:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="../specificstory.xsl"?>

<story>
     <dc:title>The Ant and His Treasure</dc:title>
     ....
     <sc:genre>folktale, animal tale, inspiration story,
         success story, trickster story
      </sc:genre>
```

```
    <sc:theme>self-reliance, determination, </sc:theme>
    <sc:tale-type>fable</sc:tale-type>
    <sc:motif>breadcrumb, weak characters</sc:motif>
    <sc:setting>anthill, nature</sc:setting>
    <sc:characters>ant, bee, cockroach, spider</sc:characters>
    <sc:archetypes>trickster animals, weak-will</sc:archetypes>
    <sc:coda>Stop crying and keep trying.</sc:coda>
   ....
</story>
```

The XML/RDF metadata allows users to search the story database by title, abstract, taletype, performer name, etc., for a digital video/audio storytelling performance, select the one they want to play, view the performance via a streaming media player, focus on relevant parts of the narrative transcript of the storytelling performance, and/or interact with the agents in a dialogue about the narrative. We have integrated access to information sources related to a given story into a natural metaphor – a virtual storytelling agent which is modeled after what people ask and answer about the story – while being aware of the ontology and the context of the story, modeled as hierarchy of classes.

The knowledge base creates an agent instance based on the class to which the story is known to belong and provides inferences about related stories and default assumptions, which are used for queries not covered by the pattern extracted from the online chat sessions. The Jinni 2003 [18] Prolog compiler's support for multiple cyclic inheritance allows stories to be organized based on multiple classification criteria, very much as if they were related Web pages linked to each other. Jinni's classes are simply Prolog files with *include* declarations. They can be located at arbitrary URLs on the Web and can inherit predicate definitions from each other. When story instances are created, the object constructor receives the URLs to the locations of the multimedia (digital video/audio) recording of the storytelling event, the story transcript, and the log of the story-related query/answering chat session.

3 An Architecture for Knowledge Intensive Conversational Agents

During the iterative design and development process of our storytelling agents we have noticed that the focus can be easily lifted towards a framework supporting conversational agents where domain specific information, lexical and semantic knowledge and common sense rules interoperate and enhance each others expressiveness. The resulting generic agent architecture is described in **Fig 1**.

We will overview the various components of the architecture and their interactions in the following sections.

4 Server Side Prolog Agents

The agent architecture is centered around Jinni 2003 [18–20] or BinProlog based [21, 22] **Prolog Server Agents** able to run as extensions of a Prolog based

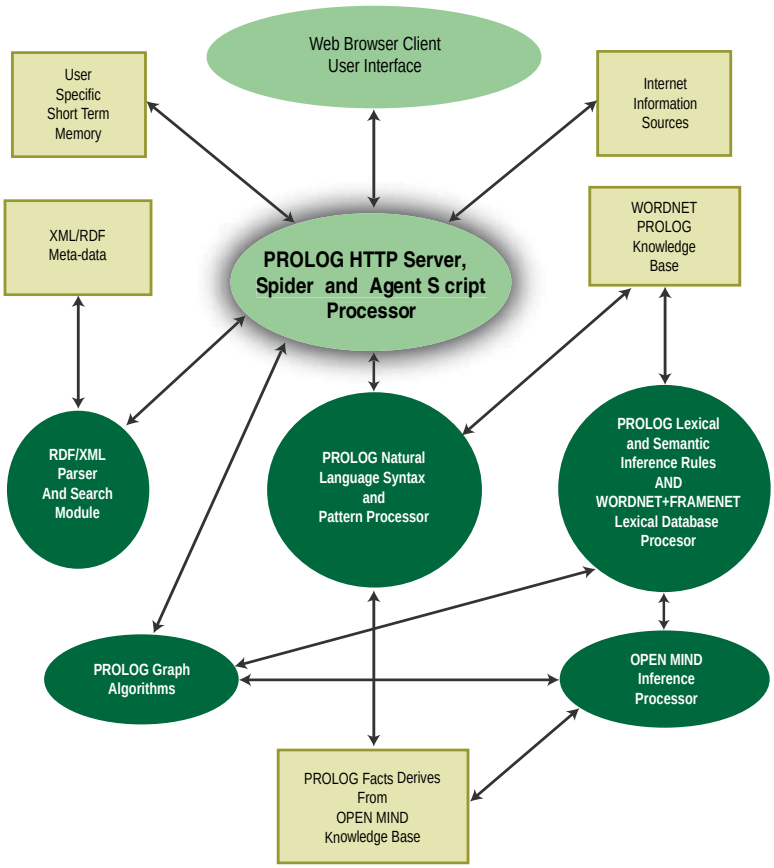

Fig. 1. A Knowledge Intensive Conversational Agent Architecture

Web server. Server agents can run on separate threads and accomplish various functions ranging from Web based information extraction to dynamic generation of Web pages. At the cost of a few hundred lines of Prolog code (mostly DCG grammar based) we also provide basic HTTP services. As a result we can deploy Jinni applets or BinProlog based Web services without having to interface to an external Web server.

5 Conversational Agents as Voice Enabled Web Services

Our Agents are deployed using Prolog Web servers and server side Prolog script processing capabilities. This provides seamless integration between the knowledge base, the shallow script processor and the XML metadata reflected as a Prolog set of story specific facts and rules. The Web components are also developed using exclusively XML/XSL/XHTML pages to ensure a natural binding of Web content to database fields and easy parsing by script processors.

We have used Microsoft Agent [23] components embedded in a dynamically generated Java Script Web page to provides easy integration of client-side voice and animation services. Under Internet Explorer, the dynamically generated Web pages trigger automatic download of the Microsoft Agent controls from the Microsoft server on first use. Client-side voice interaction is provided through the SAPI voice API's text-to-speech component. Specific text and animation commands are generated by our Prolog Server Agent processor which edits annotations made in a page template like the following:

```
<html>
  <head>
    <title>Prolog Server Agent Output</title>
  </head>
  <body language="Javascript" onLoad="OnLoad()">
    <OBJECT id=
      /* reference to Microsoft Agent downloads ... */
    </OBJECT>
    <SCRIPT language="Javascript">

var aAgent;var qAgent;var res;

function initAgent(name,url) {
  AgentControl.Characters.Load(name,url);
name = AgentControl.Characters.Character(name);
return name;
}
function initQ() {
  qAgent=initAgent("qAgent",
   "http://agent.microsoft.com//agent2//chars//peedy//peedy.acf");
}
function initA() {
  aAgent=initAgent("aAgent",
```

```
   "http://agent.microsoft.com//agent2//chars//merlin//merlin.acf");
}
function agentSpeak(agent,message) {
  agent.Get("state", "Showing, Speaking");
agent.Get("animation", "Greet, GreetReturn");
agent.Show();agent.Get("state", "Hiding");
agent.Play("Greet");
res=agent.Speak(message);
agent.Hide();
}
function OnLoad() {
  initQ();initA();

agentSpeak(qAgent,"{{?spoken_query}}");
aAgent.Wait(res);
  agentSpeak(aAgent,"{{?spoken_answer}}");
}
    </SCRIPT>
    <p>
      <font color="#000099"><b>{{?login}}:</b>{{?query}}</font>
      <br><b>agent:</b> {{?answer}}
    </p>
    <hr><b>History Window</b>
    <pre>
{{?history}}
</pre>
  </body>
</html>
```

Note the presence of the `{{?..}` patterns which will be expanded by our Prolog Server Agent processor into the actual text to be spoken by the client-side text-to-speech processor. The Pattern Processor described in the next function (also providing shallow natural language processing for queries) is used to locate the patterns and replace them on the fly with content from the Prolog database or an associative list.

6 A Definite Clause Grammar-based Pattern Processor

We have designed a generic Definite Clause Grammar based Pattern processor which works on arbitrary data (character codes, tokens, sentences) to detect and aggregate patterns at a given syntactic level.

The predicate `mach_pattern(Pattern,InputList)` matches `Pattern` against `InputList`. `Pattern` can contain any combination of constants, constrained variables of the form `X:P` where `P` is a predicate about `X`, as well as *Gap variables* which match arbitrary sequences located between constants and constrained variables. Note that constants and constrained variables match single items and function as *known* index elements in the InputList, while Gap variables collect the text to be retrieved. If patterns contain other patterns (embedded as Lists)

Fig. 2. Conversational Agents as Voice Enabled Web Services

the mechanism allows recursive application - but most of the uses of the pattern matching mechanism in our applications have involved "shallow use" i.e. recursive embedded patterns have been seldom used. The code actually handles more powerful annotations (i.e. regular expressions and disjunctive patterns) which have been proven very useful in applications like text mining and Internet content processing.

Here is an example of a rule working on a list of natural language tokens:

```
try_match(Login,Password,Ys,Os):-
  ensure_last(Ys,'?',Is),
  % What do you <know> about <life>?
  match_pattern([V:wh_word(V),do,you,
                 Verb:is_verb_phrase([Verb|_]),Obs,'?'],Is),
  !,rotate_answer(Login,Password,what_do_you(Verb,Obs,Ds),Ds),
  ensure_last(Ds,'.',Os).
```

As the answer handler `what_do_you` generates multiple answers, we are applying to it a higher order transformer `rotate_answer` which first accumulates the answers in the dynamic Prolog database (short-term-memory) and, when no more answers can be produced, rotates the answers - this is quite important to avoid boring the users with repeated answers. The parameters `Login` and `Password` uniquely identify the user allowing allocation of user-specific server side context which provides a conversational short term memory function.

7 WordNet as a Lexical Knowledge Processor

The first step in building this functionality is quick access to the semantic and lexical knowledge provided by the WordNet database [8]. This database is also available in Prolog form (see `http://www.cogsci.princeton.edu/~wn`) and is therefore ready to be used as part of a rule-based inference system. A lexicon consists of a set of word meanings and their semantic relationships. A systematic representation of the English lexicon based in psycholinguistic considerations has been put together in the database WordNet [9].

WordNet maps word forms and word meanings as a many-to-many relation. An important characteristic of WordNet is that semantic relations (hypernymy, hyponymy, synonymy, meronimy etc.) are defined in WordNet between *meanings* instead of being defined between *words or word phrases.*

Meanings are represented by *integers* (called *synsets*) associated to sets of words and word phrases collectively defining a sense element (concept, predicate or property and also usable for indexing.

So, for example, the meaning identifier (synset) `Id=100011413` maps to the following list of words and word phrases: `[[animal], [animate,being], [beast], [brute], [creature], [fauna]]`, which collectively define a common *meaning.*

7.1 An Efficient Bidirectional Word Phrase to Meaning Mapping

We have refactored the set of predicates provided by WordNet closely following the WordNet relation set (see `http://www.cogsci.princeton.edu/~wn/doc.shtml`) to support bidirectional constant time access to the set of *meanings* associated to a given word phrase (indexed by a unique head word) and for the set of word phrases and relations associated to a given (unique) meaning.

The refactored WordNet Prolog database contains the following basic binary relations.

- `w/2`: head word to meanings
- `i/2`: meaning to words
- `l/2`: meaning to meaning links (relations from the meaning to another meaning)
- `g/2`: meanings to definitions and examples
- `r/2`: reversed meaning to meaning links
- `t/2`: toplevel meanings with their syntactic roles (nouns or verbs)
- `e/2`: exception variant words to meanings
- `k/2`: known words in definitions and examples with occurrence counts
- `n/2`: new words in definitions and examples with occurrence counts

For a given word (as *humble* in the following example) the relation `w/2` returns a list of meanings:

```
?- w(humble,Meanings).
Meanings=[
  302269648,201415418,301839431,
  201414096,302162242,301551117,109699111
].
```

For a meaning, we have a number of alternative words or word phrases, with attributes. Among them, the last argument provides frequency of occurrence in a corpus of texts and will be used for disambiguation:

```
?-i(302269648,WordInfo).

WordInfo=[
  f(1,[humble],s,1,3),
  f(2,[low],s,7,0),
  f(3,[lowly],s,1,1),
  f(4,[modes],s,5,0),
  f(5,[small],s,3,155)]).
```

Links (like the sim/1 synonymy link) are collected on a list:

```
?-l(302269648,Links).
  Links=[sim(302269385)]).
```

Definitions and examples originally present in WordNet are preparsed so that they can be processed efficiently, if needed, at runtime. We also collect frequency information and word forms not present in the form of WorldNet entries.

```
?-g(302269648,DefinitionAndExamples).

DefinitionAndExamples=[
  def([low,or,inferior,in,station,or,quality]),
  ex([a,humble,cottage],
  ex([a,lowly,parish,priest]),
  ex([a,modest,man,of,the,people]),
  ex([small,beginnigs])]).
```

Note that multiple syntactic categories can be present for words like "humble" (v=verb, and a=adverb).

```
i(201415418,[f(1,[humble],v,1,1)])
...
i(301839431,[f(1,[humble],a,2,1)])
...

i(201414096,[
  f(1,[humiliate],v,1,2),
  f(2,[mortify],v,3,0),
  f(3,[chagrin],v,1,0),
  f(4,[humble],v,2,1),
  f(5,[abase],v,1,0)]).
```

Note also the presence of reversed relations like hyponyms (reverse hypernyms) and reverse meronyms. These are are precomputed to support high performance graph walk operations using BinProlog and Jinni's fast database indexing mechanisms (blackboard operations), to provide constant time access to edges related to a given node for a given relation.

We have also precomputed mappings from word variants to related meanings, based on the dictionary entry they belong.

Finally, we have precomputed "toplevel" nouns and verbs, (meaning which do not have further hypernym links).

This refactoring provides sets of facts with the following properties:

- given a head word (or a word phrase), we can extract in constant time all related information about their meanings (used in efficient parsing of textual data to lists of synsets)
- given a meaning, we can extract in constant time all relations and related meanings or words provided by the WordNet database

Overall, our refactoring simplifies WordNet while providing an efficient inference engine through Prolog rules that can digest the information contained in these basic relations.

7.2 Using Prolog Graph Algorithms

In the directed graph G of meaning nodes N_G we can see various WordNet relations as defining elements of a set of edges E_G.

Graph Walks We implement our abstraction operators as depth-K walks in such graphs. Besides the primitive WordNet relations, we use *closures*[1] as edge generators.

Making Semantic Connections through Least Common Ancestors To explore various semantic relations provided by or inferred from WordNet, we have implemented a generic graph walking algorithm which uses arbitrary relations to dynamically define *edges* and applies arbitrary conditions to the *nodes*.

Among the interesting applications to chat agents - a fast *least common ancestor* computation provides answers like in:

```
guest: Relate woman and love!
agent: After scratching my head 1 times, relating woman to
       love makes me think about girlfriend.
```

As WordNet seems to know about logic programming (the technology used in this project), the following comes out quite easily:

```
guest: Relate logic and programming.
agent: After scratching my head 1 times, relating logic to programming makes me
       think about logic programming.
```

Unfortunately, with depth limited to 3, our agent will try to save face, after not finding a connection[2]:

```
guest: Relate logic and love!
agent: About love. Love is a kind of emotion, are you
thinking about any strong feeling?
```

8 Story Abstractions

The view of stories as *programs* usable to recall and understand chains of external events suggests a technique similar to *abstract interpretation* [24] which has been used to elegantly infer program properties in the field of Programming Languages. The technique consists of lifting properties from a *concrete domain* – in this case the lexical material of a story, seen as a sequence of word phrases – to an *abstract domain* obtained by following an abstraction operator and then

[1] Our closures are predicate name+argument combinations, which receive two graph nodes as extra arguments to make-up a callable predicates.

[2] Well, the same happens at level 12 - and that's because in WorldNets view, there is no connection!

propagating back the results of the (easier) analysis from the (smaller) abstract domain to the more complex concrete domain. The idea extends to various text documents and Web documents that can be converted to or summarized as text documents.

8.1 Abstraction through the WordNet Hypernym Hierarchy

This is a simple but useful abstraction mechanism – with obvious applications to indexing. It consists of *lifting* words and word phrases extracted from the story transcripts to more general equivalents obtained by following upward links in the WordNet hypernym hierarchy – as in the following example:

```
?- lift_word(spy,1,S).
S=[[[secret,agent],
   [intelligence,officer],
   [intelligence,agent],
   [operative],...]];

no
?- lift_word(spy,2,S).
S=[[[perceiver],[observer],[beholder],[agent],...]]
];

no
?- lift_word(spy,3,S).
S=[[[person],[individual],[someone],[somebody],[mortal],[human],...]];
```

8.2 K-level Noun Abstractions

Hypernym relations are more meaningful for nouns than for other syntactic categories. Noun-related synsets form relatively deep (up to 10-12) hierarchies coming from fairly reliable common sense and natural sciences classifications. By restricting a story trace to nouns, one can get an approximation of what the story is about – at different levels of abstractions.

8.3 Verb Abstractions

Pure verb traces (obtained by selecting only verb sequences) provide an abstract view of a story's dynamics. Like Web links, WordNet graphs exhibit *small-world* structure (strong clustering and small diameter). At a deep enough level (5-10), all stories will map to sequences like `[act] [transfer] [change] [rest]` and similar verbs, organized as collections of story-independent patterns. Such patterns indicate dramatic intensity and can be used to spot out climactic points in a story.

Causality/Entailment Verb Abstractions WordNet provides a `cs(Cause, Effect)` and an `ent(Action,Consequence)` relation applying only to verbs. By using them in the context of a story, one can derive hints about what might happen next or explain why something has happened.

Answering *Why* Questions through Causality Abstractions Causal relations provide possible explanations, and as such, answers to *why* questions about a story or generate explanatory sentences usable in abstracts.

8.4 Using FrameNet Semantic Roles to Understand Conversational Context

The FrameNet corpus provides ontology data at a higher level than WordNet and allows detection of most of the semantic roles [12] relevant to the understanding of XML/RDF story-specific data as well as in detecting key elements of the conversational context. The "granularity" of FrameNet data which is described in terms of predicates (corresponding to *verbs*) and their arguments matches well our internal Prolog representations, also consisting of predicate definitions. We use a SAX based event driven parser (part of Jinni 2003) to extract only relevant data from FrameNet (a collection of a few thousand XML files of around 1000 Mb total size).

8.5 Extracting Query Answer patterns from Open Mind and chat transcripts

After simple syntactic transformations we have mapped various Open Mind and human chat transcripts to a Prolog database of "canned" question/answer facts. Through the use of noun abstractions and verb abstractions as well as synonymy relations we have significantly extended the coverage of this database.

9 Related Work

Conversational agents [25–27, 4, 28] have been identified as an effective multimodal interface element with applications ranging from user support automation to video games and interactive fiction [3, 5, 29]. Interestingly enough, conversational agents are reopening some 50 year old methodological dilemmas and challenges of artificial intelligence. We will overview them informally here, based on our own journey through various architecture and implementation decisions in building a fairly large Prolog based conversational agent in a virtual storytelling application which integrates more than a GigaByte of knowledge base data from WordNet, FrameNet and Open Mind. The distinctions stem from aspects related to *conversational intelligence* (reasoning) as well as (factual) *conversational knowledge*.

Symbolic vs. statistical inference processing This is an instance of the old AI dilemma between using logic/predicate calculus or semantic nets/conceptual graphs as a symbolic reasoning mechanism versus statistical mechanisms like Bayesian networks, genetic algorithms or artificial neural networks.

Programming vs. machine learning Should conversational agents be coded in (possibly customized) high level languages or we should use various machine learning/data mining algorithms to extract conversational intelligence from various online and offline information sources?

Hand-coded vs. automatically acquired knowledge In a way similar to the choice between hand coded and machine learned conversational intelligence, hand built conversational knowledge competes against knowledge acquired automatically by transformation/adaptation of existing knowledge repositories.

Shallow vs. deep Natural Language understanding Somewhat related to effectiveness in the "Turing test" (the ability to fool humans about an agent being or not being human) in half-serious contests like the Loebner prize. Also, it terms of conversational realism, in the context of video gaming or interactive fiction, shallow natural language processing, using a large set of patterns, often collected as an open-source, user contributed process (see [28]), has been proven a valid de facto alternative to sophisticated morphological, syntactic or semantic "deep" natural language understanding techniques, using translation of natural language to various formal representations and query processing languages.

Mimetic vs. real conversational intelligence Realistic human-like synthetic actors are now routinely used in movies and video games. The need for domain-independent conversational intelligence is particularly important in applications from interactive fiction to user support. The subconscious or explicit user expectation in all these areas is that *if an agent looks like a human and speaks like a human, then it has all the other human attributes*. This has lead chatbot implementors to a focus on "deception" techniques ranging from Eliza-style rephrasings and oracular/ambiguous/unspecific answer generation, to shifting the focus of the user (offering them a drink or talking to another agent - as seen often in video games or interactive fiction) - sometimes quite successfully in terms of psychological realism, as it is the case in Façade [30].

Knowledge intensive vs. Inference intensive AI With the advent of large, freely available lexical and common sense knowledge repositories like WordNet [8, 9] and Open Mind [16, 14] it is becoming increasingly possible to cover a large spectrum of natural language questions by finding satisfactory answers through relatively shallow (but computationally efficient) pattern matching. Interestingly, issues like "story consistency", coherent handling of the context of the conversation and its implicit assumptions require refocusing our inferential techniques towards textual rather than sentential aspects of conversational intelligence, while balancing mimetic realism and usefulness of our agents as information sources.

10 Future Work

The agent technology involved in the automation of the interactive chat query/answer patterns needs both a natural language analysis and a natural language generation component. The analysis capabilities are needed to understand the question and the generation capabilities are needed to construct the answer.

Answering *what-is-this-about* questions is relatively easy – by extraction of dominant nouns and noun phrases from each story. However, creating a dialogue to get at the deeper hermeneutics of the story or the impact of a storytelling performance narrative upon an individual is harder. Different people will select a different trace in a story to chat about. A story trace is a sequence of meanings extracted from the lexical material of a story to which one or more meaning transformations are applied. The semantic ambiguity coming from the polysemy of the lexical material is intensified by the pragmatic ambiguity of the listener's personal experience, the parameters of the storytelling performance, and the nature of the multimedia experience (seeing video, hearing audio tracks, reading a transcript, listening to a musical story, etc.)

Through the use of WorldNet, abstractions can be traced to help determine what a given story and its parts are about. WorldNet contains semantic links to allow the users to navigate on a network of meaning-to-meaning relationships. Meaning elements obtained by navigating WorldNet concept hierarchies naturally generalize the meaning of individual sentences. By starting from a story's lexical material and working upward in word meaning hierarchies to understand higher level indexing terms, story similarities and differences can be compared and query/answer patterns can be automatically extracted.

In the context of our general architecture, we have identified the following issues for the future developments of our conversational agent technology:

- Improving the methods by which agent scripts and story specific metadata are extracted from interactive query/answering transcripts.
- Further work with the story traces as a method to analyze natural language document content and ontology-driven story projections to match question patterns to relevant content parts in collections of natural language documents.
- How goal driven question generation and abductive explanations can be used in the context of mixed initiative dialogs.
- Find creative uses for the new Google meta-search API recently implemented as a Jinni 2003 extension module for XML/RDF/DAML knowledge extraction from Web sources and their use for answer generation.
- Extract more complex *conversational intelligence* from various FrameNet and Open Mind data files and take advantage of their parsed forms (to be converted from Lisp and XML to Prolog) as a mechanism for deep natural language query pattern matching.

11 Conclusion

This paper has described a logic programming based agent architecture for knowledge-intensive conversational agents deployed as self-contained Web services. The architecture uses an XML-based Web Interface, RDF based Semantic Web data, object-oriented content hierarchies and a Prolog based natural language and a knowledge processor. The architecture merges these technologies to build voice-enabled, easy to use end user applications with significant knowledge processing capabilities. The technology has applications for online teaching, user support, information retrieval, interactive fiction and video game authoring.

References

1. O. Balet, P. Kafno, F. Jordan, and T. Polichroniadis. The VISIONS Project. In *Proceedings of the International Conference on Virtual Storytelling*, Avignon, France, September 2001.
2. M. Rousseau. The Interplay between Form, Story and History. In *Proceedings of the International Conference on Virtual Storytelling*, Avignon, France, September 2001.
3. M. Zancanaro, A. Cappelletti, and C. Signorino. Interactive Storytelling: People, Stories, and Games. In *Proceedings of the International Conference on Virtual Storytelling*, Avignon, France, September 2001.
4. M. Cavassa, F. Charles, and S. Mead. Characters in Search of an Author: A.I. Based Virtual Storytelling. In *Proceedings of the International Conference on Virtual Storytelling*, Avignon, France, September 2001.
5. C. Fencott. Virtual Storytelling as Narrative Potential: Towards an Ecology of Narrative. In *Proceedings of the International Conference on Virtual Storytelling*, Avignon, France, September 2001.
6. Dan Brickley Eric Miller, Ralph Swick. Resource Description Framework (RDF). Technical report, www.w3.org, 2003. Available at http://www.w3.org/RDF/.
7. G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, and M. Scholl. RQL: A Declarative Query Language for RDF.
8. George Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. Five papers on WordNet. CSL Report 43, Cognitive Science Laboratory, Princeton University, July 1990.
9. C. Felbaum. *Wordnet, an Electronic Lexical Database for English.* Cambridge: MIT Press, 1998.
10. R. Mihalcea and D. Moldovan. eXtended Wordnet: progress report, 2001.
11. Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The Berkeley FrameNet project. In Christian Boitet and Pete Whitelock, editors, *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics*, pages 86–90, San Francisco, California, 1998. Morgan Kaufmann Publishers.
12. D. Gildea and D. Jurafsky. Automatic labeling of semantic roles, 2002.
13. Nancy Chang, Srini Narayanan, and Miriam R. L. Petruck. From Frames to Inference, 2002.
14. Push Singh. The Open Mind Common Sense. Technical report, M.I.T Media Lab, 2003. http://commonsense.media.mit.edu.

15. Push Singh. The Open Mind Common Sense project. Technical report, KurzweilAI.net, 2002. http://www.kurzweilai.net/meme/frame.html?main=/articles/art0371.html.
16. Push Singh. The public acquisition of commonsense knowledge. In *Proceedings of AAAI Spring Symposium on Acquiring (and Using) Linguistic (and World) Knowledge for Information Access*, Palo Alto, California, 2002. AAAI.
17. Paul Tarau, Koen De Boschere, Veronica Dahl, and Stephen Rochefort. LogiMOO: an Extensible Multi-User Virtual World with Natural Language Control. *Journal of Logic Programming*, 38(3):331–353, March 1999.
18. BinNet Corporation. The Jinni 2003 Prolog Compiler: a High Performance Java and .NET based Prolog for Object and Agent Oriented Internet Programming. Technical report, BinNet Corp., 2003. Available at http://www.binnetcorp.com/download/jinnidemo/JinniUserGuide.html.
19. Paul Tarau. Fluents: A Refactoring of Prolog for Uniform Reflection and Interoperation with External Objects. In John Lloyd, editor, *Proceedings of CL'2000*, London, July 2000. LNCS, Springer-Verlag.
20. Paul Tarau. Inference and Computation Mobility with Jinni. In K.R. Apt, V.W. Marek, and M. Truszczynski, editors, *The Logic Programming Paradigm: a 25 Year Perspective*, pages 33–48. Springer, 1999. ISBN 3-540-65463-1.
21. Paul Tarau and Veronica Dahl. High-Level Networking with Mobile Code and First Order AND-Continuations. *Theory and Practice of Logic Programming*, 1(1), March 2001. Cambridge University Press.
22. Paul Tarau and Veronica Dahl. A Logic Programming Infrastructure for Internet Programming. In M. J. Wooldridge and M. Veloso, editors, *Artificial Intelligence Today – Recent Trends and Developments*, pages 431–456. Springer, LNAI 1600, 1999. ISBN 3-540-66428-9.
23. Microsoft. The Microsoft Agent Home Page. http://www.microsoft.com/msagent.
24. P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *4th ACM Symp. Principles of Programming Languages*, pages 238–278, 1977.
25. E. Andr'e, T. Rist, S. van Mulken, M. Klesen, and S. Baldes. The automated design of believable dialogue for animated presentation teams, 2000.
26. S. van Mulken, E. Andr, and J. Muller. The persona effect: How substantial is it, 1998.
27. Mark G. Core and James F. Allen. Coding dialogues with the DAMSL annotation scheme. In David Traum, editor, *Working Notes: AAAI Fall Symposium on Communicative Action in Humans and Machines*, pages 28–35, Menlo Park, California, 1997. American Association for Artificial Intelligence.
28. A.L.I.C.E. AI foundation. Artificial Intelligence Markup Language (AIML). Technical report, A.L.I.C.E. AI foundation, 2001. Available at http://alice.sunlitsurf.com/TR/2001/WD-aiml/.
29. Elizabeth Figa and Paul Tarau. The VISTA Project: An Agent Architecture for Virtual Interactive Storytelling. In N. Braun and U. Spierling, editors, *TIDSE'2003*, Darmstadt, Germany, March 2003.
30. Michal Mateas and Andrew Stern. Integrating Plot, Character and Natural Language Processing in the Interactive Drama Facade. In N. Braun and U. Spierling, editors, *Proceedings of the Technologies for Interactive Digital Storytelling and Entertainment Conference*, Darmstadt, Germany, March 2003.