Hybrid Natural Language Processing in a Customer-Care Environment

David Reitter*, Stefan Covaci*, Florin Oltean**, Catalin Bacanu**, Traian Serbanuta**

Abstract

CyMON is an industrial platform for one-to-one customer care applications. It compromises facilities to implement user-friendly web-sites with an emotional, learning interface. The CyMON Natural Language Processing Engine can understand written input and react accordingly. It is based on hybrid linguistic and statistical algorithms to analyze natural language input. Each sentence is assigned an appropriate contextualized meaning representation. This paper describes the results of an evaluation of basic pattern matching techniques and argues in favor of their augmentation with more sophisticated models taken from linguistics and statistics. The natural language processing techniques applied in CyMON are described in detail. An overview of related work is provided.

1 Introduction

Agentscape's natural language processing modules address a broad variety of user interaction: chatting, informing, data mining and queries to product databases. These tasks contribute essentially to the functionality of portals, company sites and e-business web sites. These application areas are subsumed by the term Customer Relationship Management (CRM). Whenever an organization wishes to provide support to existing customers, raise their value or acquire new ones via the Internet or other interactive channels, natural language processing (NLP) is the key for an easy-to-use interface. The CyMON-NLP components incorporate the technology to understand general written input and quickly generate knowledge bases to support the implementation of an CRM system in a specific domain.

The NLP modules have been integrated into two platforms: Cyb (Create Your Bot) and CyMON (Create your match and organizing netware). Cyb is a personal assistant application which runs on a user's desktop and assists him in doing recurring tasks and in organizing his documents. CyMON is an agent-based platform for one-to-one CRM applications. CyMON includes features such as implicit and explicit profiling (data-mining), profile-matching, session-tracking, context-driven, reactive and proactive, emotional-driven agent behavior.

2 Initial prototypes

Two major implementations of the predecessor platform Si.MON have been made. One, www.flirtmaschine.de, is an interactive, web community system with regularly updated content. It provides user profiling and a profile-driven matching engine. The topics of Flirtmaschine relate to a virtual flirt and partner-finding platform and are supported by journalistic life-style content.

*Agentscape AG, Berlin, Germany , [d.reitter, s.covaci]@agentscape.de

**Agentscape SRL, Bucharest, Romania, [olf, cba, tse]@agentscape.ro

The second application was an anonymous session-based informational web site for a major German fashion company (*www.puc-online.de*). A virtual shop assistant reacts to natural-language (textual) input and gives advice on, e.g., what kind of clothes to wear, how to remove stains or how to bind a tie. The functionality includes profiling and generation of reports based on given market-survey questions.

The natural language understanding component in Si.MON works with 3000 ordered pattern matching rules. These reference user profile data and the current context (such as the topic of the page the user was currently viewing). The rules check the input sentence for the occurence of text patterns (preconditions), which were defined separately. A total of 10 000 such patterns were defined. Patterns could contain references to other patterns and also disjunctions thereof. The matching algorithm evaluated each rule separately and stopped on the first hit. The system was implemented in Java.

2.1 Critical evaluation of the prototypes

A good basis for an evaluation of the initial pattern-matching approach was given by the log files from the Flirtmaschine project. Around 30.000 distinct input sentences and the system responses were gathered.

A manual quality-oriented evaluation grouped the question-answer pairs in three categories: 1) Good. These answers were convincing and implied no hints on the artificial dialog situation. 2) Acceptable. The system did not seem to understand the question, but gave a reasonable, maybe pro-active answer in the situation. 3) Wrong. The answer was clearly wrong according to the judgement of the testers. We counted 54 percent of good and acceptable answers. There were two prominent reasons for wrong answers.

Over-Generation. The recognition and interaction rules were formulated too lose in many occasions. This applied to regular phenomena, such as the scope of negation, which could have been ruled out if some more elaborate form of syntactic and semantic analysis had been used. A wildcard operator matched all characters except word-separators. It was used to cover suffixes with formulations like "H\#us\#" (matching Haus, Hauses, Häuser, Häusern..., German for: house}) turned out to match too many lexical items.

Missing semantic coverage. This was a much smaller problem, because in many cases, when a user question was not forseen in a specific rule, a more general rule applied. After all, this led to a lot of acceptable, but not good answers.

Missing lexical coverage. The Si.MON NLP patterns were built manually. Of course, a lot of synonyms and semantically related words were missing. The round trip time between the user pushing the enter button to send a question and the system displaying the answer on a client machine with a 64K internet connection was estimated in the range of several seconds. (The log files did not contain any time stamp information.)

The manpower needed to assemble the natural language rules and patterns in Flirtmaschine amounted to 6 man months.

Analyzing the corpus itself, we found that the complexity of the sentences entered into the system was rather low. In fact, many of the input sentences were not grammatical according to standard German. Referential expressions and ellipsis constructs were widely used. We found spelling errors and missing capitalization.

A similar, but more superficial analysis has been done with another corpus gathered via our Peek&Cloppenburg application. It yielded much more domain-specific questions. The sentences were longer, more detailed and more often grammatically correct in comparison to the chat-oriented Flirtmaschine. This underlines the assumption that in serious customer relationship management applications, users tend to be more explicit and formally correct in their questions.

From these results and from the business-level plans regarding the CyB/CyMON platform, we enumerated the following requirements for the new NLP component.

Good precision (correctness). The answers of the system should obviously be adequate in an increased number of cases.

Good recall (domain coverage). The NLP component must provide means to configure it with reasonable efforts to cover typical aspects of a customer relationship management domain. This includes product, support and company information, product-specific user profiling, analytical marketing.

By choosing a tagger which could deal with unknown words, we extended the mechanisms to react to unknown input. A more efficient grammar writing process could also contribute to broader coverage. Efficient language engineering. Administrators are usually not trained linguists, but employees of public relations departments and trainees. They pursue a practical approach to language and have acquired the target language as mother tongue. An initial training time of a maximum of three days was expected to be acceptable in order to learn the main features of the system. To speed up the development, the target user must be able to select and integrate pre-assembled knowledge components. Linguistic knowledge should be reusable.

This requirement was met by adding:

- a stemming component,
- the CyMON Development Kit, a comfortable visual tool for managing and extending the semantic grammar,
- a clearer grammar format and compositional semantics.
- On the generation-side of the system, we chose to implement a unification-based selection of textual reactions.

Multilingual Support. Domain-specific and language-specific capabilities need to be kept separate. The system has to be extensible in a modular way by means of language packages that provide basic support for different languages. By separating the concerns and by clearly defining formats for different databases, this requirement was met. We started with German and English as initial target languages.

Domain Extendibility. Domain knowledge should be easily extended. The system should be able to incorporate further recognition and dialog resources. A separation between recognition module and reaction rules with a semantic representation connecting both led to increased independence of the two kinds of data and their administration. The CyMON Development Kit also supports importing data into the semantic grammar. Robustness. Variations in orthography, such as capital letters, missing or incorrect interpunctuation and spelling mistakes should be ignored to a certain extent. The system should pursue strategies to give convincing answers even to input with unknown words or phrases.

To rise to this expectation, Edit-Distance Algorithms to loosen up pattern matching were integrated as well as normalization of diacritics and capitalization.

Run-time efficiency and scalability. The system is employed in the CyMON platform. As scale it must handle 2000 simultaneously connected users. Furthermore, several precautions had to be taken to be able to downgrade the NLP system in order to integrate it also in a desktop-application with low memory usage and sufficiently fast response times.

To tackle the requirement on the response-time, we implemented the NLP component in C++ using advanced parsing optimization algorithms.

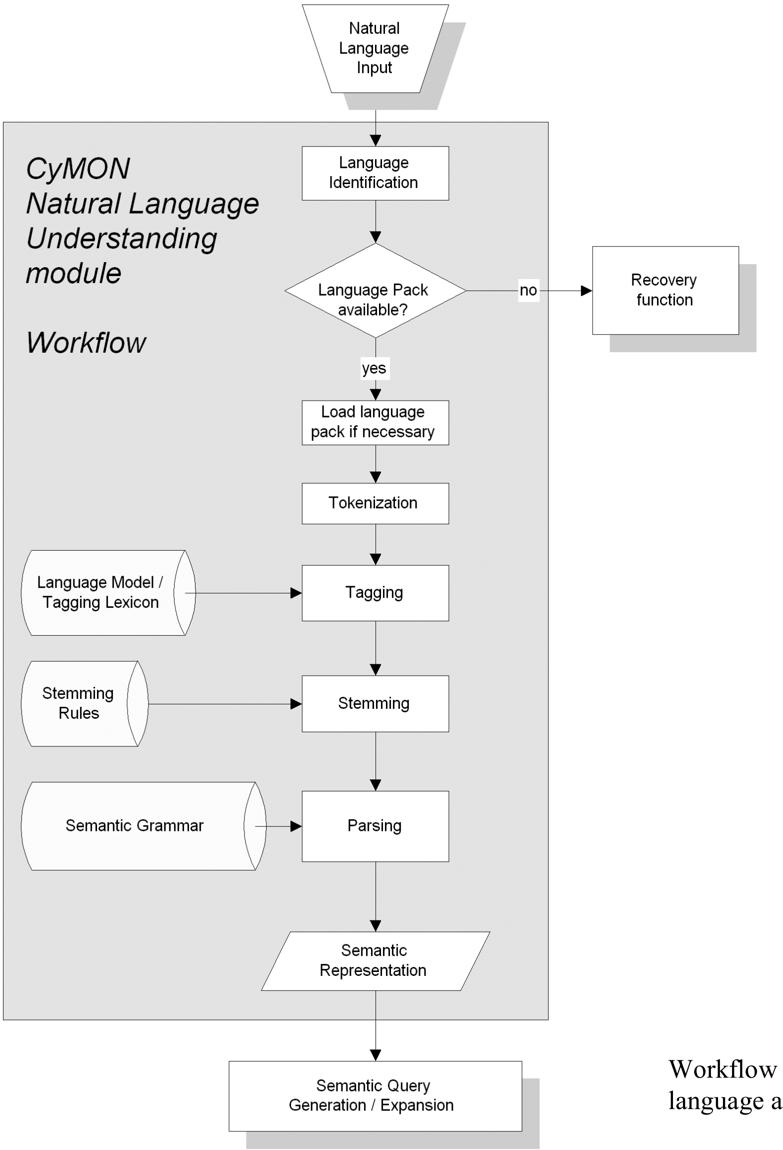
Application Areas. Within the domain of customer relationship management applications, the NLP had to address two different kinds of tasks: Domain-specific question-answering, Chatting and data-mining. For chatting, an increased robustness is needed, while for question-answering, a more detailed analysis of the input question is required.

3 Language technology system architecture

There are two groups of components in CyMON-NLP. The Natural Language Understanding (NLU) component is responsible for analyzing an input sentence. It returns a semantic representation of the input. The natural language generation component - the Dialog Manager selects an appropriate answer dialog from a dialog database to a given semantic stimulus. Other components such as the Interaction Engine take care of steering the whole system.

To configure the CyMON-NLP system, the following resources have to be supplied:

- Domain-dependent resources: A semantically oriented grammar describing phrases and sentences to be recognized, and a
- set of dialogs. These files adhere to the XML standard.
- Language-dependent resources: Lexica and language models for tagging and stemming services. Statistical language models for the language identification.
- A central configuration file.



Workflow diagram of the CyMON language analysis component

gehörte	ADJA (0.2)	VVFIN (0.8)	
bewegt	ADJD (0.17)	VVFIN (0.5)	VVPP (0.33)
weniger	ADV (0.66)	PIAT (0.21)	PIS (0.13)

Table 1: Entries from a tagger lexicon. The codes in the table denote, for each word form, an ambiguity class of morphosyntactic tags. E.g. 'bewegt' may by tagged as predicative adjective (ADJD), finite full verb (VVFIN) or past participle (VVPP). The numbers in parantheses denote the relative probability of each tag within its ambiguity class.

CyMON-NLU follows a hybrid approach. We integrated morpho-syntactic disambiguation and a lemmatization (stemming) module into a semantically oriented grammar. We were able to convert much of the patterns and rules written in the Flirtmaschine project. Most of them serve as basis for the development of new domain knowledge bases.

3.1 Multi-linguality: language detection

Before the morpho-syntactic analysis is started, a statistical language recognizer tries to identify the language used in the natural language input. It compares parameters such as average word length and consonants used with language models gained from test corpora. The NLU is instantiated for each detected language. The CyMON-NLU can be configured to recognize the language once per session or for every input.

3.2 Morpho-syntactic disambiguation: the tagging service

We decided to assign morpho-syntactic tags to each word of the input for two reasons. Ambiguity problems arose whenever the grammar rules were not very constrained. Many top-level rules of the grammar contain immediate-dominance, but not linear-precedence rules, because the word order of German (the language we started with) varies on regular and irregular bases. Constraining these rules with syntactic tags could help to decrease parsing errors and increase parsing speed. Another very important rationale for introducing the tagging were unknown words in the input. A statistical tagger (here: "guesser") can assign tags even to unknown words by looking at their context and, in some cases, at hints from their suffixes.

We implemented a solution inspired by the algorithms described in br00. Brants describes algorithms for a powerful statistical part-of-speech tagger. He argues that his parser, based on Markov models, performs better than other approaches, such as finite-state, rule-based or memory-based taggers.

According to his data, the Markov models yield even better results than more general statistical Maximum Entropy algorithms.

As a tagset we chose the EAGLES-compliant Stuttgart-Tuebingen tagset STTS, described in [Schiller 1995]. This tagset contains 54 tags, grouped in 11 classes.

We will shortly outline the data structures used to tag data as described in [Brants 2000]. The algorithm needs, aside of the tokenized text, two data structures as external resources. The first is a lexicon containing full forms of words and their ambiguity class. This ambiguity class denotes the list of tags the words can assume in a syntactic configuration. Furthermore, for each tag in an ambiguity class of a specific word, its relative probability is stored. This gives the probability a certain word is found, given a certain tag is encountered ($P(w_1|t_1)$ with w_1 being a word and t_1 being a tag). Table 1 shows an example.

The second data structure contains probabilities of the different contexts the tags occur in. The tagging algorithm needs information on unigram ($P(t_1)$), bigram ($P(t_2|t_1)$) and trigram ($P(t_3|t_1,t_2)$ with t_1, t_2, t_3 being tags occurring sequentially to words in the corpus) probabilities. This is stored in the language model. Table 2 shows an example.

The lexicon can be augmented independently of the language models without retraining the corpus, provided the ambiguity classes of the words are known. In this case, the lexical tag probabilities must be set to a default value.

The guessing part of the tagger relies on a third data structure: probability information regarding tag assignments. Tags are determined based on suffix detection. The data structures for the tagger are obtained by a training process, where a (manually) tagged corpus is given. In simple terms, the system 'learns', what common orders of tags are in a sentence of a specific language. In our case, we used the Flirtmaschine corpus (*www.flirtmaschine.de*) to train the tagger. After an initial tagging phase using different language models, the results had to be validated and corrected manually. Out of 133.000 words contained in the Flirtmaschine corpus, we corrected 4.300 and trained the tagger. The lexicon was augmented with bigger sources and currently contains around 64.600 entries. The language model contains probabilities for 8270 distinct trigrams. The final accuracy of the tagger is higher than 96 percent.

We plan to add further entries to the lexicon. The way we want to do this is to automatically assign ambiguity classes to the words by looking at the their suffixes. Because this is essentially the same algorithm as the one the guessing module of the tagger uses, this certainly will not improve the tagging performance. For this reason, additional words will be validated manually.

3.3 Ignoring morphologic alternations: the stemming service

A brief look into the pattern definitions of the original Flirtmaschine implementation yielded an efficiency problem.

Inflected wordforms that are semantically equivalent in the given context, had to be dealt with somehow. The # wildcard operator described above led to over- and under-generation, depending on how it was employed by the language engineer.

In order to ignore inflection markers, users can rely on a stemming algorithm. It strips suffixes off from words. Because the stemmer will only normalize words for the purpose of comparison with a predefined pattern, the stemming service does not need to generate a linguistically proper stem. The implementation of this functionality relies on the tagger, since the suffixes employed depend - among other factors - on the P.O.S. category of the word. This information is retrieved from the tagging lexicon or calculated by the guesser.

As a resource, a list of rewriting rules is available for each tag class. A rewriting rule is a search and replace pair. The search terms are regular expressions with the replace terms referencing parts of the matched string. The rewriting rules are applied in their configured order. They are executed immediately. A feeding relationship of rules is desired, i.e. an earlier rule may provide the context for a later one. Table stemmrex gives an excerpt from our rewriting rules for German. In the phrase *größere Autos* (larger cars), they normalize the adjective to its base form *groß*. For irregular forms, a lexicon is queried.

	frequency	tag sequence
unigram	4752	VAFIN
bigram	145	VAFIN - NN
trigram	15	VAFIN - NN - ART

Table 2: Excerpt from a language model for n-gram HMM tagging. A frequency measure is given for each possible tag sequence. A normalized probability value can easily be derived from the frequencies.

```
# the P.O.S. tags the following rules apply to
   [ADJA] [ADJD]

# Genus/Case/Number markers
   (.*[aeiou].*)(e|er|en|em|es) -->   \1

# Comparative / Superlative markers
   (.*[aeiou].*)(er|rer|st)  -->   \1

# Remove Umlauts
   ä --> a
   ö --> o
   ü --> u
```

Some simple search&replace regular expressions used to stem German adjectives.

3.4 Identifying phrase and sentence meanings: grammars

To calculate a semantic representation from a sequence of tagged and stemmed tokens, a parser has to find the proper combination of patterns. As mentioned earlier, CyMON.NLP works with a two-level grammar.

The first level of rules is a list of productions with decreasing priority. These productions, called *combinations* are immediate-dominance (ID) rules. This means, they enumerate a list of other rules (second level rules, pattern) that have to be contained in the constituent described. They may also make linear-precedence (LP) statements on the order of rule appearance. Furthermore, negation is possible. We have four pattern operators that can be evaluated in the combinations:

Contain: a given pattern must be contained in the input.

Match: the given pattern must match the input exactly.

NotContain,NotMatch: Negations of contain and match.

Expressions with these operators may be combined using the AND (conjunction) and OR (disjunction) boolean operators. It is easy to see that AND combined with *Contain* leads to ID rules, while AND combined with *Match* is an ID/LP statement.

The second level grammar rules are called *patterns*. These patterns are meant to work at the phrasal and lexical level. They may reference other patterns and/or raw (terminal) text. Operators used at this level are concatenation (*Concat*) and disjunction (*Any*). Usually, concatenation is used on the phrasal level; disjunction is meant to group semantically equivalent words or patterns. Additionally, patterns make use of the *Morph* operation, which will license morphological alternation of the words and patterns. The parser will compare only the stems of the words.

The CyMON-NLU parser can be categorized as a top-down look-ahead chart parser. While evaluating each combination in their respective order, a chart is built with information on the results of pattern checks. This prevents pattern rules to be analyzed twice. To further optimize the parsing process, an index is consulted. This index contains all known lexical words and, for each word, a list those combinations which reference the word. Thus, each word appearing in the input triggers the check of a set of combinations.

With this data-driven technique, the parser can rule out in the beginning most of combinations. However, closed-class words (functional categories such as prepositions, auxiliaries, determiners) had to be excluded from indexing, because they appeared too frequently. This restriction raised no problems in practice. Each combination must contain at least one positive condition on a word that is not of a functional category. Another restriction is applied to combinations that make use of wildcard operators. Those combinations must contain at least one non-functional word. Again, this restriction is not a problem with real life test sentences.

The interpretation of misspelled words is another problem we were facing. We implemented an algorithm to calculate a minimal edit distance, which is a sum of all insertions, deletions and changes of single characters necessary to match the original word and the user input. The resulting value is normalized to the word length (which effectively allows more spelling errors to occur in longer words). Furthermore, anticipating that spelling errors occur more often towards the end of a word, we modified the algorithm to use the position of the character in the word as a weighting factor when counting the changes.

The edit distance challenges our word index, because misspelled words will not be found in the index. An alternative solution we are investigating is to calculate a Soundex-like checksum for each word and use this code has a hash key for the index.

Obviously, the parsing part is the most time-consuming operation of the language analysis system. It analyzes an average of 7-9 sentences per second (on a Pentium III/600 desktop machine) with our hard disk based grammar database. Running through all combinations (with indexing turned off) costs about 13 seconds.

3.5 Building semantic representations

CyMON-NLU semantic representations consist of three classes of elements: atoms, propositions and semantic terms.

Atoms These are atomic items identifying a concept or an instance in our universe of discourse, such as “car”, “roadster”, “bmw318i”. They depend largely on the domain CyMON-NLU is configured for. In a network-based knowledge description, atoms could refer to a concept class or concept instance node.

How are atomic expressions assigned? In a semantically oriented grammar, a production rule is used to specify the syntactical construction of a phrase and its semantically equivalent alternatives. That is why atomic expressions are assigned at the phrasal level, i.e. along with a pattern or part of a pattern.

In a pattern, one or more semantic variables may be assigned an atomic value. This is how CyMON-NLU allows to assign more than one atomic identifier to each pattern. This method should not be mistaken for a chance to introduce alternative meanings in case of semantic ambiguities, i.e. homonyms. Instead, it is used to structure the meaning of a phrase.

For example, we may include four patterns:

- Pattern 1 matches all expressions for a BMW 318.
  Its semantics contain the assignment: prod-id:= *bmw318*
- Pattern 2 matches all expressions for a Honda Civic.
  Semantics: prod-id:= *hondaCivic*
- Pattern 3 matches all expressions for a roadster.
  Semantics: prod-type:= *roadster*
- Pattern 4 contains a disjunction of Patterns 1, 2.
  Semantics: prod-type:= *closedcar*
- Pattern 5 contains a disjunction of Patterns 3 and 4.
  Semantics: prod:= *car*

After all, each product description has three levels of specialization. Depending on the context it appears in, the grammar may refer one of the levels. With this mechanism, a hierarchy of concept classes and concept instances can be build. Even if useful in other applications, an external semantic network is not necessary at this level.

A slightly different use case of the multi-value phrasal semantics would be to select several slots specifying properties of the object described. After all, the meaning of a phrase is by no means necessarily atomic. However, not all users of the CyMON-NLU will build up a hierarchy like that. Instead, they will rather assign only one atomic meaning to each phrase.

Propositions. Different phrasal semantics are combined into a single proposition. This is done at the first level of grammar productions, i.e. in combinations. Propositions describe a speech act. For the semantic representation of a proposition, we use a language-independent structure that contains a semantic predicate and a set of slot-filler pairs.

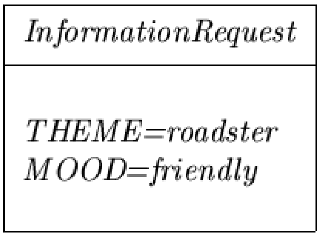
Values filling the slots can be atoms or strings.

In an attribute-value matrix (known from unification-based formalisms), each name-value pair should be considered a restriction. Depending on the predicate, certain pieces of information should occur in the structure. The predicate specifies the type of the structure. Depending on the predicate, there is a number of mandatory and optional arguments. In the construction of a proposition, these arguments (slots) are filled with values from semantic variables, i.e. with the object identifiers from the phrases of a sentence. Alternatively, direct assertion of a value is possible. The assignment of strings is discussed later on.

A typical semantic representation for the sentence

Könnten Sie mir bitte ein paar Cabrios zeigen?

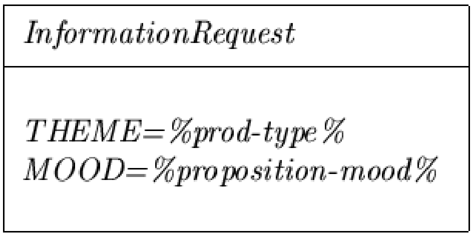
looks like this:



InformationRequest

THEME=roadster
MOOD=friendly

While the value of *THEME* is mandatory, *MOOD* can be left out if there is no information present in the sentence. Both arguments are contributed from the semantic content of the patterns. So, the template constructing this semantic representation is formulated like this:



InformationRequest

THEME=%prod-type%
MOOD=%proposition-mood%

CyMON-NLU gives language engineers a high degree of freedom to design their own semantic system. In the accompanying tutorials, we encourage the use of compositional elements whenever possible. We propose a small set of predicates along with a standardized set of arguments, oriented towards theta roles in linguistics. However, creating a combination for each single proposition is possible. Arguments may be avoided; a huge number of distinct predicates will result. Even though such atomic propositions are easier to understand at a first glance, this will hinder language engineers to reuse and share parts of their grammars.

Semantic terms. Input sentences can contain more than one proposition. Consider the input

I bought this washing machine the other day, and now it is loosing water!

Essentially, it contains two propositions: The customer bought a washing machine, and: it is defect, in particular, it has a leak.

To represent these sentences, the complete semantic representation of a sentence is a semantic term, which is a conjunction of propositions. Other elements known from formal semantics, such as the disjunction operator or quantifiers have not been necessary in our current applications. Also, they would ask for proper model-checking when finding an appropriate reaction (see below). The disjunctive semantics can be evaluated as follows: the system will evaluate each of the propositions and combine all actions resulting from each single proposition.

Compared to classical methods of representing meaning, one notices that the CyMON terms make use of ideas from predicate logic, but restrict it in many ways. The most important difference is that our semantic structures are flat. Thus, this method will not do when representing semantically complex propositions. Nevertheless it covers a wide range of utterances, and probably all we will need in customer care domains. Even more importantly, it has proven to be simple enough in order to be used by non-trained language engineers. The same applies to the way of building up semantics. While compositional approaches, such as Montague's Semantics ([Montague 1974]) or Discourse Representation Theory ([Kamp/Reyle 1993]) provide a maximum of re-usability, the way modelling is done always calls for trained computational linguists and works a quite restricted input language.

3.6 Processing unknown input

If an unknown word occurs in a relevant position of the input, it will be matched by a restricted wildcard. The word is stored in the semantic representation as a value of string type. Thus, the semantic representation is not restricted to contain only semantic concepts.

User input: Where can I find a(n) +UnknownProduct+/NN?

The wildcard expression in this sentence will match one or more words tagged NN and store the phrase in a variable named UnknownProduct. This variable can be referred to lateron when constructing the semantic representation:

UserRequestsUnknown
THEME= %UnknownProduct%

So, along with the known semantic content, a phrasal content can be stored. Even though the system does not know the 'semantic' content of a phrase, it is still able to identify its syntactic function and thus restrict the morphological variation. We expect the user to enter mainly grammatical phrases. If a neologism is entered here, the system will use it in its answer as well.

This allows for easy construction of ELIZA-like dialogs. For example, a convincing answer for the situation described above would be

Answer: Unfortunately, we don’t carry %UnknownProduct%. But I can present you a list of product categories to choose from.

Problems arise if the morpho-syntactic features of the unknown word cannot be guessed. This is a clear limitation of the system which could be ruled out with a more detailed. First investigations about this have been made in [Tufis 2000].

3.7 Reaction-finding / generation

The search for a correct reaction is, similar to the ordered evaluation of grammar rules, an iterative process. The CyMON Dialog Manager tries to compare the given semantic representation of the input with a basic semantic representation attached to a possible output reaction. The check is done for all reactions in a list; the first matching one is accepted.

The comparison of both semantic representations, that of the input and the one aligned with the output, is defined as a subsumption relationship. An input representation Sem_Q is subsumed an reaction representation Sem_A, iff

- The predicates Sem_Q and Sem_A are identical
- Each argument of Sem_A is present in Sem_Q and has the same value in both representations.

This subsumption check assures that additional information may always be added in the analysis grammar; so adding information to the query will expand the search space. In contrast, the representations attached to system reactions are restrictive. So, there may be reaction conditions of different specificness. As in the semantic grammar, the reactions with the specific conditions are put first in the list.

Depending on the interaction model, a reaction is defined by a set of further statements, which are executed by the state-machine reasoning engine. Alternatively, if the neural network is being used, the triggering semantic representation will activate a network node.

For chat-like conversations, a reaction can be given immediately. For database enquiries, the respective interaction engine will carry out a predefined sequence of actions. It will access the arguments from the semantic representation.

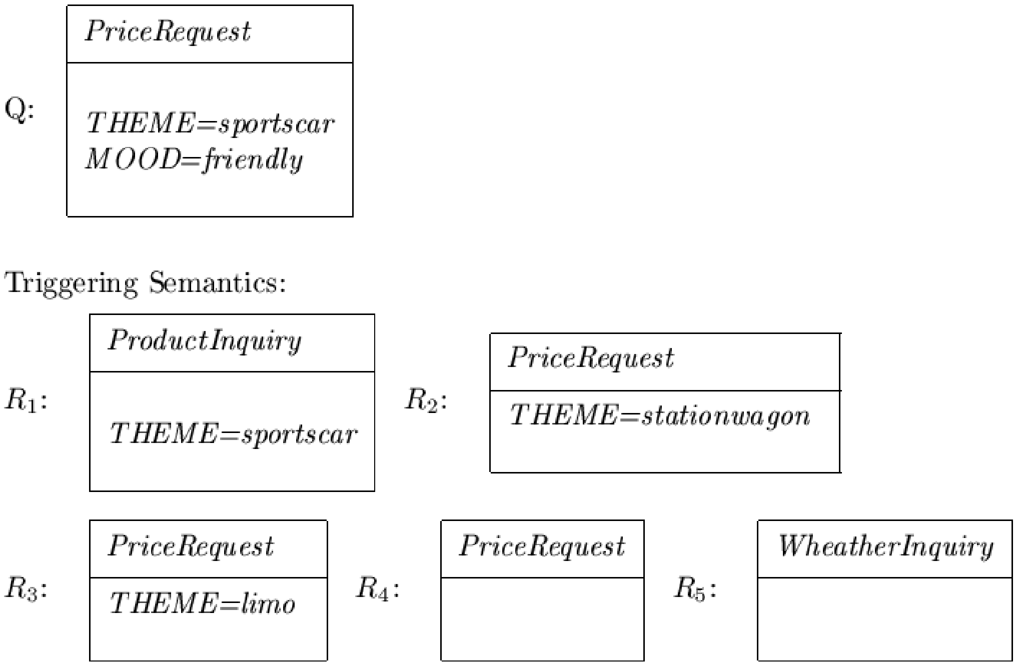


Figure 3: For given input semantics, an appropriate reaction is searched. The first satisfied subsuming reaction condition is R_4.

4 CyMON Development Kit

4.1 Grammar editor

Users can create the grammars by means of a graphical user-interface (GUI). It provides a hierarchy to group and structure production rules. The modular architecture of the CyMON system is mirrored in the Development Kit. Language-specific data are kept in ready-to-use components outside of the model. The external modules compromise tagging models and the stemming algorithms.

The domain-specific semantic grammars can be edited directly in the CDK. A special extension was created to make knowledge extension as intuitive as possible. A linguistic assistant asks the user to input a sample sentence. This is analyzed using a tagger and a nominal phrase extractor. Known grammar rules which apply to parts of the input are recognized. Ambiguities in the analysis are ruled out manually. For new parts of the input, extra grammar productions are created automatically.

All components of the system can be configured through a central configuration file.

5 Evaluation of CyMON.NLU

Regarding engineering effort, we verified that an implementation of a test scenario could be done in about 4 man/weeks. The stemmer led to a significant decrease in the time needed to enter wordforms; the compositional semantics proved advantages in reusability of patterns. An evaluation of the new runtime system is still under way. Functional tests of the system will, however, always be run on an instantiation of the technology, i.e. a domain-specific knowledge base. To a great extend, the recognition capabilities relate to the knowledge base.

6 Related work

Alicebot ([Wallace 1999], *alicebot.org*) specifies an open Markup Language, called AIML. It works with stimulus-response rules, which include pattern-based natural language analysis. Some morphological alternations are integrated, as well as rewriting-techniques. Regarding natural language, the specification has a focus on dialog tracking. ALICE has been implemented and used in various demo applications.

Among the industrial projects, Artificial Life's WebGuide (*www.artificial-life.com*) is a a web-based configurable smart bot for natural language-based user interactions. It can understand natural language input and integrates emotions.

Autonomy's Kenjin (*www.autonomy.com*) is a client-side assistance software that proactively learns and proposes relevant content to the user. It has the ability to link the user's personal knowledge into a dedicated community to find users with similar interests. Autonomy's approach is purely statistical. The Portal-in-a-Box package is a related portal platform with knowledge management capabilities.

The Cyc product family (from Cycorp, *www.cyc.com*) is based on a big multi-contextual knowledge base, designed to capture a large portion of what we normally consider consensus knowledge about the world. At the same time it provides an inference engine, a set of interface tools, and a number of special-purpose application modules supporting context-based knowledge management and querying. A dedicated script language is specified as the knowledge representation language with Cyc. The Cyc products contribute to the creation of intelligent bots.

Network Query Language (*www.nqli.com*) is an interpreted scripting language that enables rapid development of bots, spiders and intelligent agents. It provides learning, but no NLP.

Novomind IQ (*www.novomind.de*) is a development platform for customer care agents. According to the company's presentations, the modular architecture should enable the construction of individually tailored personal agents for different application contexts.

Vista's Virtual Friends (*www.virtual-friends.com*) are avatars designed to assist customers in their shopping activities on a specific web site. The technology has primitive natural language interfaces (spoken/written).

KiwiLogic (*www.kiwilogic.de*) offers a Windows-based bot construction kit with a simple, pattern-matching based Natural-Language-Understanding system.

Several other "light" bots, which reside on a user's desktop, are available for low prices, such as BonziBuddy (entertaining agent with internet/calendar utilities, *www.bonzi.com*) or Agentscape's Cyb (multi-purpose, extendible personal desktop assistant, *www.cybs-garage.de*).

7 Further work

A replacement for the grammar-based analysis engine is currently in development. It will provide means to find answers in relational databases using graph-theoretic operations on a knowledge network. Input preprocessing will be done with the existing tagger and a rule-based NP chunker.

8 Conclusions

We have shown how industrial natural language analysis and lingware development can be improved significantly with an augmentation of pattern matching techniques by parsing with a simplified semantic grammar, statistical tagging algorithms and rule-based stemming functions. We have demonstrated a semantic representation formalism that can be used by untrained language engineers. Finally, we have described a unification-based answer finding technique with template-based natural language generation.

We believe that linguistic analysis techniques (both rule-based and statistical) provide means to overcome some of the deficits of pattern-matching based dialog systems.

Acknowledgements

The CyMON-NLU project has been made possible by Agentscape AG, Berlin. The implementation and unit test of its components were carried out by Catalin Bacanu, Catalin Iercosan, Florin Oltean and their team at Agentscape S.R.L., Bucharest. Dan Tufis, RACAII Bucharest, supported us greatly in tagging techniques.

References

[Abney 1996] Abney, Steven: *Statistical Methods and Linguistics*. In: Judith Klavans and Philip Resnik (eds.), The Balancing Act. The MIT Press, Cambridge, MA. 1996

[Brants 2000] Brants, Thorsten: *TnT - a statistical part of speech tagger*. In: Proceedings of he sixth applied natural language processing conference ANLP-2000, Seattle 2000.

[Kamp/Reyle 1993] Kamp, Hans / Reyle, Uwe: From Discourse to Logic: Introduction to Model-theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory. Computational Linguistics 21(2):265-268.

[Montague 1974] Montague, Richard (1974). Formal Philosophy: Selected Papers of Richard Montague. Edited with an introduction by R. H. Thomason. Yale University Press.

[Schiller 1995] Schiller, Anne / Teufel, Simone / Stöckert, Christiane: *Vorläufige Guidelines für das Tagging deutscher Textcorpora mit STTS.* Seminar für Sprachwissenschaft (Draft), Tübingen 1995.

[Tufis 2000] Tufis, Dan: *Using a large set of EAGLES-compliant morpho-syntactic descriptors as a tagset for probabilistic tagging. I*n: Proceedings of the International Conference on language resources and evaluation LREC '2000, Athens, 2000. pp. 1105-1112

[Volk 1998] Volk, Martin / Schneider, Gerold: *Comparing a statistical and a rule-based tagger for German.* In: Proc. of Konvens-98, Bonn 1998

[Wallace 1999] Richard Wallace: *The A.L.I.C.E. nexus*, 1999. At http://www.alicebot.org/

[Wothke 1993] Wothke, Klaus / Weck-Ulm, I. / Heinecke, J. / Mertineit, O. / Pachunke, T. : *Statistically-Based Automatic Tagging of German Text Corpora with Part-of-Speech -- Some experiments.* IBM TR 75.9302, 1993.