Growing Semantic Grammars

Marsal Gavaldà and Alex Waibel
Interactive Systems Laboratories
Carnegie Mellon University
Pittsburgh, PA 15213, U.S.A.
marsal@cs.cmu.edu

Abstract

A critical path in the development of natural language understanding (NLU) modules lies in the difficulty of defining a mapping from words to semantics: Usually it takes in the order of years of highly-skilled labor to develop a semantic mapping, e.g., in the form of a semantic grammar, that is comprehensive enough for a given domain. Yet, due to the very nature of human language, such mappings invariably fail to achieve full coverage on unseen data. Acknowledging the impossibility of stating a priori all the surface forms by which a concept can be expressed, we present GSG: an empathic computer system for the rapid deployment of NLU front-ends and their dynamic customization by non-expert end-users. Given a new domain for which an NLU front-end is to be developed, two stages are involved. In the *authoring* stage, GSG aids the *developer* in the construction of a simple domain model and a kernel analysis grammar. Then, in the *run-time* stage, GSG provides the *end-user* with an interactive environment in which the kernel grammar is dynamically extended. Three learning methods are employed in the acquisition of semantic mappings from unseen data: (*i*) parser predictions, (*ii*) hidden understanding model, and (*iii*) end-user paraphrases. A baseline version of GSG has been implemented and preliminary experiments show promising results.

1 Introduction

The mapping between words and semantics, be it in the form of a semantic grammar,[1] or of a set of rules that transform syntax trees onto, say, a frame-slot structure, is one of the major bottlenecks in the development of natural language understanding (NLU) systems. A parser will work for any domain but the semantic mapping is domain-dependent. Even after the domain model has been established, the daunting task of trying to come up with all the possible surface forms by which each concept can be expressed, still lies ahead. Writing such mappings takes in the order of years, can only be performed by qualified humans (usually computational linguists) and yet the final result is often fragile and non-adaptive.

Following a radically different philosophy, we propose rapid (in the order of days) deployment of NLU modules for new domains with on-need basis learning: let the semantic grammar grow automatically when and where it is needed.

2 Grammar development

If we analyze the traditional method of developing a semantic grammar for a new domain, we find that the following stages are involved.

1. *Data collection.* Naturally-occurring data from the domain at hand are collected.
2. *Design of the domain model.* A hierarchical structuring of the relevant concepts in the domain is built in the form of an ontology or domain model.
3. *Development of a kernel grammar.* A grammar that covers a small subset of the collected data is constructed.
4. *Expansion of grammar coverage.* Lengthy, arduous task of developing the grammar to extend its coverage over the collected data and beyond.
5. *Deployment.* Release of the final grammar for the application at hand.

The GSG system described in this paper aids all but the first of these stages: For the second stage, we have built a simple editor to design and analize the Domain Model; for the third, a semi-automated way of constructing the Kernel Grammar; for the fourth, an interactive environment in which new semantic mappings are dynamically acquired. As for the fifth (deployment), it advances one place: after the short initial authoring phase (stages 2 and 3 above) the final application can already be launched, since the semantic grammar will be extended, at run-time, by the non-expert end-user.

3 System architecture

As depicted in Fig. 1, GSG is composed of the following modules: the Domain Model Editor and the

[1] Semantic grammars are grammars whose non-terminals correspond to semantic concepts (e.g., `[greeting]` or `[suggest_time]`) rather than to syntactic constituents (such as `Verb` or `NounPhrase`). They have the advantage that the semantics of a sentence can be directly read off its parse tree, and the disadvantage that a new grammar must be developed for each domain.

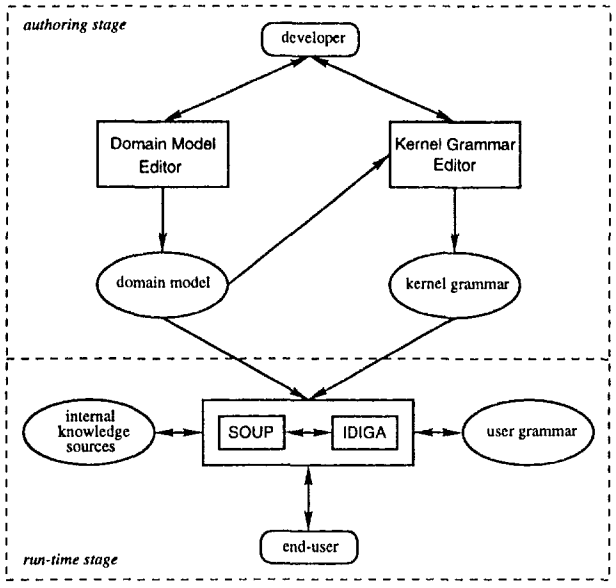

Figure 1: System architecture of GSG.

Kernel Grammar Editor, for the authoring stage, and the SOUP parser and the IDIGA environment, for the run-time stage.

3.1 Authoring stage

In the authoring stage, a developer[2] creates the Domain Model (DM) with the aid of the DM Editor. In our present formalism, the DM is simply a directed acyclic graph in which the vertices correspond to concept-labels and the edges indicate concept-subconcept relations (see Fig. 2 for an example).

Once the DM is defined, the Kernel Grammar Editor drives the development of the Kernel Grammar by querying the developer to instantiate into grammar rules the rule templates derived from the DM. For instance, in the DM in Fig. 2, given that concept `[suggest_time]` requires subconcept `[time]`, the rule template `[suggest_time]` ⟵ `[time]` is generated, which the developer can instantiate into, say, rule (2) in Fig. 3.

The Kernel Grammar Editor follows a concrete-to-abstract ordering of the concepts obtained via a topological sort of the DM to query the developer, after which the Kernel Grammar is *complete*[3] and

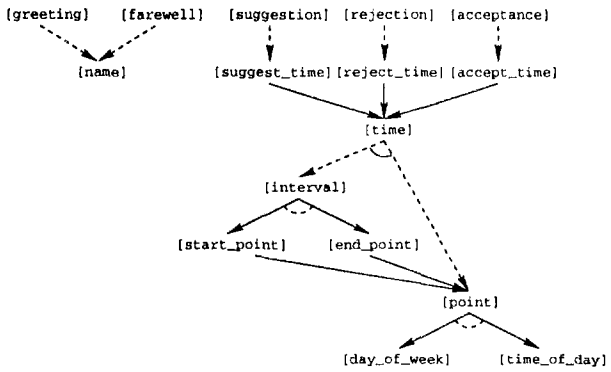

Figure 2: Fragment of a domain model for a scheduling task. A dashed edge indicates *optional subconcept* (default is *required*), a dashed angle indicates *inclusive subconcepts* (default is *exclusive*).

(1) `[suggestion]` ⟵ `[suggest_time]`
(2) `[suggest_time]` ⟵ *how about* `[time]`
(3) `[time]` ⟵ `[point]`
(4) `[point]` ⟵ **on* `[day_of_week]` **[time_of_day]*
(5) `[day_of_week]` ⟵ *Tuesday*
(6) `[time_of_day]` ⟵ *afternoon*

Figure 3: Fragment of a grammar for a scheduling task. A '*' indicates optionality.

the NLU front-end is ready to be deployed.

It is assumed that: (*i*) after the authoring stage the DM is fixed, and (*ii*) the communicative goal of the end-user is expressible in the domain.

3.2 Run-time stage

Instead of attempting "universal coverage" we rather accept the fact that one can never know all the surface forms by which the concepts in the domain can be expressed. What GSG provides in the run-time stage are mechanisms that allow a non-expert end-user to "teach" the meaning of new expressions.

The tight coupling between the SOUP parser[4] and the IDIGA[5] environment allows for a rapid and multi-faceted analysis of the input string. If the parse, or rather, the paraphrase automatically generated by GSG[6], is deemed incorrect by the end-user, a learning episode ensues.

[2]Understood here as a qualified person (e.g., knowledge engineer or software developer) who is familiar with the domain at hand and has access to some sample sentences that the NLU front-end is supposed to understand.

[3]We say that grammar G is *complete* with respect to domain model DM if and only if for each arc from concept i to concept j in DM there is at least one grammar rule headed by concept i that contains concept j. This ensures that any idea expressible in DM has a surface form, or, seen it from another angle, that any in-domain utterance has a paraphrase that is covered by G.

[4]Very fast, stochastic top-down chart parser developed by the first author incorporating heuristics to, in this order, maximize coverage, minimize tree complexity and maximize tree probability.

[5]Acronym for *interactive, distributed, incremental grammar acquisition.*

[6]In order for all the interactions with the end-user to be performed in natural language only, a generation grammar is needed to transform semantic representations into surface forms. To that effect GSG is able to cleverly use the analysis grammar in "reverse."

By bringing to bear *contextual constraints*, GSG can make predictions as to what a sequence of unparsed words might mean, thereby exhibiting an "empathic" behavior toward the end-user. To this aim, three different learning methods are employed: parser predictions, hidden understanding model, and end-user paraphrases.

3.2.1 Learning

Similar to Lehman (1989), learning in GSG takes place by the dynamic creation of grammar rules that capture the meaning of unseen expressions, and by the subsequent update of the stochastic models. Acquiring a new mapping from an unparsed sequence of words onto its desired semantic representation involves the following steps.

1. *Hypothesis formation and filtering.* Given the context of the sentence at hand, GSG constructs hypotheses in the form of parse trees that cover the unparsed sequence, discards those hypotheses that are not approved by the DM[7] and ranks the remaining by likelihood.
2. *Interaction with the end-user.* The ranked hypotheses are presented to the end-user in the form of questions about, or rephrases of, the original utterance.
3. *Dynamic rule creation.* If the end-user is satisfied with one of the options, a new grammar rule is dynamically created and becomes part of the end-user's grammar until further notice. Each new rule is annotated with the learning episode that gave rise to it, including end-user ID, time stamp, and a counter that will keep track of how many times the new rule fires in successful parses.[8]

3.2.2 Parser predictions

As suggested by Kiyono and Tsujii (1993), one can make use of parse failures to acquire new knowledge, both about the nature of the unparsed words and about the inadequacy of the existing grammar rules. GSG uses *incomplete parses* to predict what can come next (i.e. *after* the partially-parsed sequence in left-to-right parsing, or *before* the partially-parsed sequence in right-to-left parsing). This allows two kinds of grammar acquisition:

1. *Discovery of expression equivalence.* E.g., with the grammar in Fig. 3 and input sentence *What about Tuesday afternoon?* GSG is able to ask the end-user whether the utterance means the same as *How about Tuesday afternoon?* (See Figs. 4, 5 and 6). That is because in the process of parsing *What about Tuesday afternoon?* right-to-left, the parser has been able to match rule (2) in Fig. 2 up to *about*, and thus it hypothesizes the equivalence of *what* and *how* since that would allow the parse to complete.[9]
2. *Discovery of an ISA relation.* Similarly, from input sentence *How about noon?* GSG is able to predict, in left-to-right parsing, that *noon* is a `[time]`.

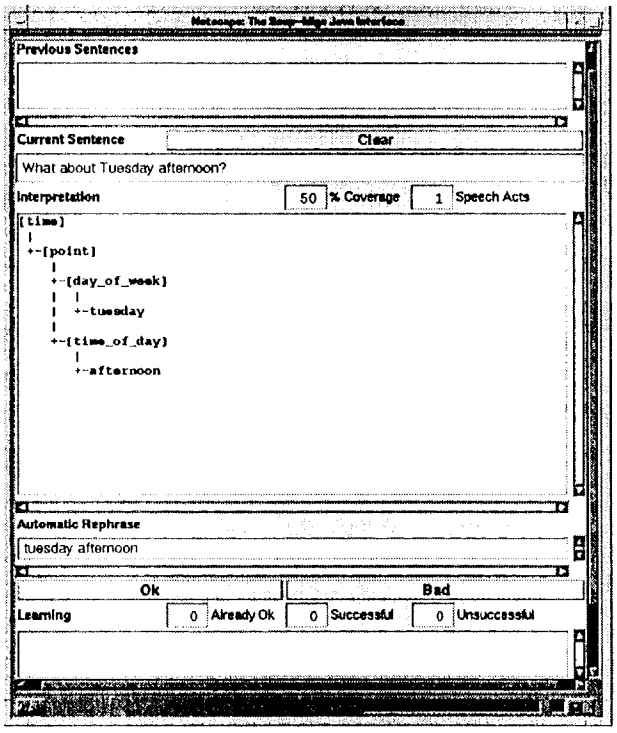

Figure 4: Example of a learning episode using parser predictions. Initially only the temporal expression is understood...

3.2.3 Hidden understanding model

As another way of bringing contextual information to bear in the process of predicting the meaning

[7] I.e., parse trees containing concept-subconcept relations that are inconsistent with the stipulations of the DM.

[8] The degree of generalization or *level of abstraction* that a new rule should exhibit is an open question but currently a Principle of Maximal Abstraction is followed:

(a) Parse the lexical items of the new rule's right-hand-side with all concepts granted top-level status, i.e., able to stand at the root of a parse tree.

(b) If a word is not covered by any tree, take it *as is* into the final right-hand side. Else, take the root of the parse tree with largest span; if tie, prefer the root that ranks higher in the DM.

For example, with the DM in Fig. 2 and the grammar in Fig. 3, *What about Tuesday?* is abstracted to the maximally general *what about* `[time]` (as opposed to *what about* `[day_of_week]` or *what about* `[point]`).

[9] For real-world grammars, of, say, over 1000 rules, it is necessary to bound the number of partial parses by enforcing a maximum beam size at the left-hand side level, i.e., placing a limit on the number of subparses under each nonterminal to curb the exponential explosion.

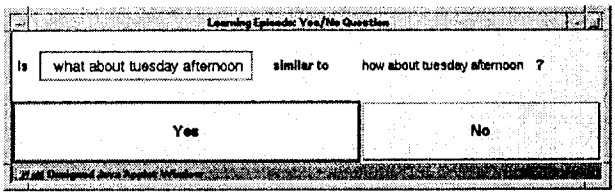

Figure 5: ...but a correct prediction is made...

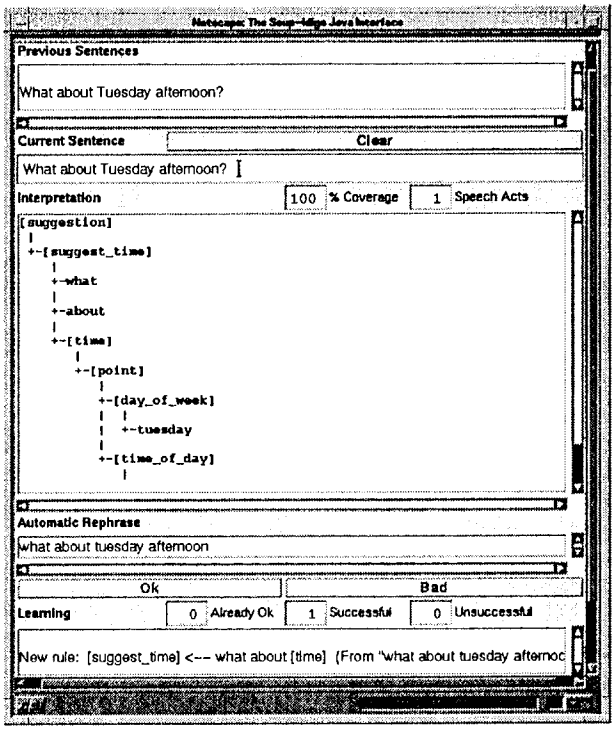

Figure 6: ...and a new rule is acquired.

of unparsed words, the following stochastic models, inspired in Miller et al. (1994) and Seneff (1992), and collectively referred to as hidden understanding model (HUM), are employed.

- *Speech-act n-gram.* Top-level concepts can be seen as speech acts of the domain. For instance, in the DM in Fig. 2 top-level concepts such as `[greeting]`, `[farewell]` or `[suggestion]`, correspond to discourse speech acts, and in normally-occurring conversation, they follow a distribution that is clearly non-uniform.[10]
- *Concept-subconcept HMM.* Discrete hidden Markov model in which the states correspond to the concepts in the DM (i.e., equivalent to grammar non-terminals) and the observations to the embedded concepts appearing as immediate daughters of the state in a parse tree. For example, the parse tree in Fig. 4 contains the following set of <state, observation> pairs: {<`[time]`, `[point]`>, <`[point]`, `[day_of_week]`>, <`[point]`, `[time_of_day]`>}.
- *Concept-word HMM.* Discrete hidden Markov model in which the states correspond to the concepts in the DM and the observations to the embedded lexical items (i.e., grammar terminals) appearing as immediate daughters of the state in a parse tree. For example, the parse tree in Fig. 4 contains the pairs: {<`[day_of_week]`, *tuesday*>, <`[time_of_day]`, *afternoon*>}.

The HUM thus attempts to capture the recurring patterns of the language used in the domain in an *asynchronous* mode, i.e., independent of word order (as opposed to parser predictions that heavily depend on word order). Its aim is, again, to provide predictive power at run-time: upon encountering an unparsable expression, the HUM hypothesizes possible intended meanings in the form of a ranked list of the most likely parse trees, given the current state in the discourse, the subparses for the expression and the lexical items present in the expression.

Its parameters can be best estimated through training over a given corpus of correct parses, but in order not to compromise our established goal of rapid deployment, we employ the following techniques.

1. In the absence of a training corpus, the HUM parameters are seeded from the Kernel Grammar itself.
2. Training is maintained at run-time through dynamic updates of all model parameters after each utterance and learning episode.

3.2.4 End-user paraphrases

If the end-user is not satisfied with the hypotheses presented by the parser predictions or the HUM, a third learning method is triggered: learning from a paraphrase of the original utterance, given also by the end-user. Assuming the paraphrase is understood,[11] GSG updates the grammar in such a fashion so that the semantics of the first sentence are equivalent to those of the paraphrase.[12]

[10] Needless to say, speech-act transition distributions are empirically estimated, but, intuitively, the sequence <`[greeting]`, `[suggestion]`> is more likely than the sequence <`[greeting]`, `[farewell]`>.

[11] Precisely, the requirement that the grammar be *complete* (see note 3) ensures the existence of a suitable paraphrase for any utterance expressible in the domain. In practice, however, it may take too many attempts to find an appropriate paraphrase. Currently, if the first paraphrase is not understood, no further requests are made.

[12] Presently, the root of the paraphrase's parse tree directly becomes the left-hand-side of the new rule.

	Perfect	Ok	Bad
Expert before	55.41	17.58	27.01
Expert after	75.68	10.81	13.51
Δ	*+20.27*	*−6.77*	*−13.50*
End-user$_1$ before	58.11	18.92	22.97
End-user$_1$ after	64.86	22.97	12.17
Δ	*+6.75*	*+4.05*	*−10.80*
End-user$_2$ before	41.89	16.22	41.89
End-user$_2$ after	48.64	28.38	22.98
Δ	*+6.75*	*+12.16*	*−18.91*

Table 1: Comparison of parse grades (in %). Expert using traditional method vs. non-experts using GSG.

4 Preliminary results

We have conducted a series of preliminary experiments in different languages (English, German and Chinese) and domains (scheduling, travel reservations). We present here the results for an experiment involving the comparison of expert vs. non-expert grammar development on a spontaneous travel reservation task in English. The grammar had been developed over the course of three months by a full-time expert grammar writer and the experiment consisted in having this expert develop on an unseen set of 72 sentences using the traditional environment and asking two non-expert users[13] to "teach" GSG the meaning of the same 72 sentences through interactions with the system. Table 1 compares the correct parses before and after development.

It took the expert 15 minutes to add 8 rules and reduce bad coverage from 27.01% to 13.51%. As for the non-experts, end-user$_1$, starting with a similar grammar, reduced bad parses from 22.97% to 12.17% through a 30-minute session[14] with GSG that gave rise to 8 new rules; end-user$_2$, starting with the smallest possible *complete* grammar, reduced bad parses from 41.89% to 22.98% through a 35-minute session[14] that triggered the creation of 17 new rules.

60% of the learning episodes were successful, with an average number of questions of 2.91. The unsuccessful learning episodes had an average number of questions of 6.19 and their failure is mostly due to unsuccessful paraphrases.

As for the nature of the acquired rules, they differ in that the expert makes use of optional and repeatable tokens, an expressive power not currently available to GSG. On the other hand this lack of generality can be compensated by the Principle of Maximal Abstraction (see note 8). As an example, to cover the new construction *And your last name?*, the expert chose to create the rule:

`[request_name]` ⟵ `*`*and your last name*

whereas both end-user$_1$ and end-user$_2$ induced the automatic acquisition of the rule:

`[request_name]` ⟵ `CONJ POSS [last]` *name*.[15]

5 Discussion

Although preliminary and limited in scope, these results are encouraging and suggest that grammar development by non-experts through GSG is indeed possible and cost-effective. It can take the non-expert twice as long as the expert to go through a set of sentences, but the main point is that it is possible at all for a user with no background in computer science or linguistics to teach GSG the meaning of new expressions without being aware of the underlying machinery.

Potential applications of GSG are many, most notably a very fast development of NLU components for a variety of tasks including speech recognition and NL interfaces. Also, the IDIGA environment enhances the usability of any system or application that incorporates it, for the end-users are able to easily "teach the computer" their individual language patterns and preferences.

Current and future work includes further development of the learning methods and their integration, design of a rule-merging mechanism, comparison of individual vs. collective grammars, distributed grammar development over the World Wide Web, and integration of GSG's run-time stage into the JANUS speech recognition system (Lavie et al. 1997).

Acknowledgements

The work reported in this paper was funded in part by a grant from ATR Interpreting Telecommunications Research Laboratories of Japan.

References

Kiyono, Masaki and Jun-ichi Tsujii. 1993. "Linguistic knowledge acquisition from parsing failures." In *Proceedings of the 6th Conference of the European Chapter of the ACL.*

Lavie, Alon, Alex Waibel, Lori Levin, Michael Finke, Donna Gates, Marsal Gavaldà, Torsten Zeppenfeld, and Puming Zhan. 1997. "JANUS III: speech-to-speech translation in multiple languages." In *Proceedings of ICASSP-97.*

Lehman, Jill Fain. 1989. *Adaptive parsing: Self-extending natural language interfaces.* Ph.D. dissertation, School of Computer Science, Carnegie Mellon University.

Miller, Scott, Robert Bobrow, Robert Ingria, and Richard Schwartz. 1994. "Hidden understanding models of natural language." In *Proceedings of ACL-94.*

Seneff, Stephanie. 1992. "TINA: a natural language system for spoken language applications." In *Computational Linguistics*, vol. 18, no. 1, pp. 61–83.

[13] Undergraduate students not majoring in computer science or linguistics.

[14] Including a 5-minute introduction.

[15] Uppercased nonterminals (such as `CONJ` and `POSS`) are more syntactical in nature and do not depend on the DM.

Resum

Un dels camins crítics en el desenvolupament de mòduls de comprensió del llenguatge natural passa per la dificultat de definir la funció que assigna, a una seqüència de mots, la representació semàntica desitjada. Els mètodes tradicionals per definir aquesta correspondència requereixen l'esforç de lingüistes computacionals, que dediquen mesos o àdhuc anys construint, per exemple, una gramàtica semàntica (formalisme en el qual els símbols no terminals de la gramàtica corresponen directament als conceptes del domini de l'aplicació determinada), i, tanmateix, degut precisament a la pròpia natura del llenguatge humà, la gramàtica resultant mai no és capaç de cobrir tots els mots i expressions que ocorren naturalment al domini en qüestió.

Reconeixent per tant la impossibilitat d'establir a priori totes les formes superficials amb què un concepte pot ser expressat, presentem en aquest treball GSG: un sistema computacional empàtic per al ràpid desplegament de mòduls de comprensió del llenguatge natural i llur adaptació dinàmica a les particularitats i preferències d'usuaris finals inexperts.

El procés de construcció d'un mòdul de comprensió del llenguatge natural per a un nou domini pot ser dividit en dues parts. Primerament, durant la fase de *composició*, GSG ajuda el desenvolupador expert en l'estructuració dels conceptes del domini (ontologia) i en l'establiment d'una gramàtica minimal. Tot seguit, durant la fase d'*execució*, GSG forneix l'usuari final inexpert d'un medi interactiu en què la gramàtica és augmentada dinàmicament.

Tres mètodes d'aprenentatge automàtic són utilitzats en l'adquisició de regles gramaticals a partir de noves frases i construccions: (*i*) prediccions de l'analitzador (GSG empra anàlisis incompletes per conjecturar quins mots poden aparèixer tant *després* de l'arbre d'anàlisi incomplet, en anàlisi d'esquerra a dreta, com *abans* de l'arbre d'anàlisi incomplet, en anàlisi de dreta a esquerra), (*ii*) cadenes de Markov (mètodes estocàstics que modelen, independentment de l'ordre dels mots, la distribució dels conceptes i llurs transicions, emprats per calcular el concepte global més probable donats un context i uns arbres d'anàlisi parcials determinats), i (*iii*) paràfrasis (emprades per assignar llur representació semàntica a la frase original).

Hem implementat una primera versió de GSG i els resultats obtinguts, per bé que preliminars, són ben encoratjadors car demostren que un usuari inexpert pot "ensenyar" a GSG el significat de noves expressions i causar una extensió de la gramàtica comparable a la d'un expert.

Actualment estem treballant en la millora dels mètodes automàtics d'aprenentatge i llur integració, en el disseny d'un mecanisme de combinació automàtica de regles gramaticals, en la comparació de gramàtiques individuals amb gramàtiques col·lectives, en el desenvolupament distribuït de gramàtiques a través de la World Wide Web, i en la integració de la fase d'execució de GSG en el sistema de reconeixement de la parla i traducció automàtica JANUS.

摘要

设计自然语言理解系统的主要难点在于定义从词到语义的映射：通常给一个特定领域设计一个足够全面的语义语法（一种非终结符直接对应语义单元的语法）要花费一个专业人员几年的时间。而且由于自然语言天生的复杂性，即使这样的映射也不可能覆盖所有可能的情况。

由于不可能逐一列举一个概念所能对应的各种表达方式，我们提出 GSG：一种可以很快生成自然语言理解系统，并使该系统动态地适应非专家用户的语言习惯的电脑系统。

针对一个特定的领域，自然语言理解系统的形成分为两步。在设计阶段，GSG 辅助设计人员生成一个简单的领域模型和一个核心语法；在运行阶段，GSG 给用户提供一个交互式界面来动态扩展核心语法。

GSG 采用三种学习机制从未知数据中学习语义映射：语法分析器预测，隐理解模型，和用户诠释。

我们实现了一个GSG 原型，初步实验得到了较好的结果。