GRAMMAR LEARNING FOR SPOKEN LANGUAGE UNDERSTANDING

Ye-Yi Wang and Alex Acero
Microsoft Research
Redmond, Washington 98052, USA

ABSTRACT

Many state-of-the-art conversational systems use semantic-based robust understanding and manually derived grammars, a very time-consuming and error-prone process. This paper describes a machine-aided grammar authoring system that enables a programmer to rapidly develop a high quality grammar for conversational systems. This is achieved with a combination of domain-specific semantics, a library grammar, syntactic constraints and a small amount of example sentences that have been semantically annotated. Our experiments show that the learned semantic grammars consistently outperform manually authored grammars requiring much less authoring load.

1. INTRODUCTION

Semantic-based robust understanding technology has been widely used in human/machine [1-3] and human/human [4] conversational systems. It has been used by many labs in evaluating the DARPA-sponsored Airline Travel Information System (ATIS) evaluation. Such implementations have relied on manual development of a domain-specific grammar, a task that is time-consuming, error-prone and requires a significant amount of expertise. If conversational systems are to be mainstream, it becomes apparent that writing domain-specific grammars is a major obstacle for a typical application developer. Recently researchers have been working on tools for rapid development of mixed-initiative systems [5], but without addressing the problem of grammar authoring *per se*. Also, other researchers have developed tools that let a end user refine an existing grammar [6], which still relies on an initial grammar and also assumes that the developer has a good knowledge of language structures.

Automatic grammar inference has attracted the attention of researchers for many years [7-11], though most of that work has focused on toy problems. Application of such approaches on grammar structure learning for natural language has not been satisfactory for natural language understanding applications [12, 13]. This limited success is due to the complexity of the problem: that available data will typically be sparse relative to the complexity of the target grammar, and there is not a good generalization mechanism to correctly cover a large variety of language constructions.

So instead of an ambitious empirical automatic grammar inference, we focus in this paper on an engineering approach that could greatly ease grammar development by taking advantage of many different sources of prior information. In doing so, a good quality semantic grammar can be derived semi-automatically with a small amount of data.

2. SEMANTIC GRAMMAR AND MULTIPLE INFORMATION SOURCES

A semantic context free grammar (CFG), like a syntactic CFG, defines the legal combination of individual words into constituents and constituents into sentences. In addition, it also has to define the concepts and their relations in a specific domain. It is this additional dimension of variation that makes it necessary to develop a grammar for every new domain. While it is not realistic to empirically learn structures from a large corpus due to technical and resource constraints, we can greatly facilitate grammar development by integrating different information sources to semi-automatically induce language structures. These various information sources are described in this section.

2.1 Domain-Specific Semantic Information

We use semantic *schema* to define the entity relations of a specific domain in our multi-modal research. Semantic schema is used for many different purposes. It serves as the specification for a language-enabled application: once a semantic schema is defined, grammar and application logic development can proceed simultaneously according to the semantic schema. It also plays a critical role in dialog management [14]. It is also language independent in the sense that it does not specify the linguistic expressions used to express the concepts. Because of this, it is used not only for language-enabled applications, but also for integrating inputs from multi-modalities, such as mouse click events. A developer has to author the semantic schema anyway in a multi-modal application; therefore it is not an extra burden for grammar learning. Because it is language-independent, it is easy for a developer with good knowledge of an application to author its semantic schema. For the calendar domain in [16], the schema contains 16 concepts with fewer than 60 slots. This is two orders of magnitude lower than the ~3000 CFG rules for ~1000 nonterminals, so the semantic schema was developed in a couple of hours. Below is an example of concept definitions in a semantic schema:

```
<entity type="ExistingAppt" name="ApptByAttributes">
    <slot type="People"/>
</entity>
<command type="ApptUpdate" name="AddAttendee">
    <slot type="People"/>
    <slot type="ExistingAppt"/>
</command>
```

Here the semantic class ApptByAttributes is an entity with one slot that represents the attendees of an existing appointment. It covers expressions like "the meeting with Alex." (The example is greatly simplified for the sake of brevity). It has the semantic type ExistingAppt, which means that it is one of the many different ways to refer to an existing appointment. Other ways include ApptByAnaphora (e.g. "that meeting"). The semantic class AddAttendee is a command. It has two slots, which simply states that you can add People to an ExistingAppt.

2.2 Grammar Library

Some low level semantic entities, such as date, time, duration, postal address, currency, numbers, percentage, etc, are not domain-specific. They are universal building blocks that can be written once and then shared by many applications. Grammar libraries can greatly save development time.

2.3 Annotation

We can also get developers involved to annotate the data against the schema. For example, the sentence "invite Ed to the meeting with Alex" is annotated against the schema as follows:

```
<AddAttendee text="invite Ed to the meeting with Alex">
    <ApptByAttributes text="the meeting with Alex">
        <People text="Alex"/>
    </ApptByAttributes>
    <People text="Ed"/>
</AddAttendee>
```

Fig. 1. Semantic annotation of a sentence against the schema.

In Fig. 1, each XML tag is the name of a semantic class in the schema, and the sub-structures are the semantic objects that fill the slots of a semantic class. The annotation is surface-structure independent --- different sentences that convey the same meaning would have the same annotation. The use of the grammar library also eases the annotation process since we do not have to annotate to the very bottom level of concepts.

2.4 Syntactic Constraints

Domain specific language must comply with the syntactic constraints of language. Some simple syntactic clues, for example, part-of-speech constraints, are used to reduce the search space in grammar learning.

3. LEARNING PROCEDURE

3.1 Inherit Semantic Constraints from Schema

An assumption we made is that the linguistic constraints that guide the integration of smaller units into a larger chunk is an invariant for the subset of the natural language used in human-computer interaction. The things that vary are the domain-specific semantics and linguistic expressions for concepts. This allows us to create a template CFG that inherits the semantic constraints from the semantic schema. For example, the two concepts in the previous example can be automatically translated to the following template CFG:

```
<T_ExistingAppt> ➔ <C_ApptByAttributes>                          (1)
<C_ApptByAttributes> ➔ {<ApptByAttributeMods>}
   <ApptByAttributeHead> {<ApptByAttributeProperties>}         (2)
<ApptByAttributeProperties> ➔ <ApptByAttributeProperty>
   {<ApptByAttributeProperties>}                                (3)
<ApptByAttributeProperty> ➔
   <ApptByAttributePeopleProperty> |
   <ApptByAttributeStartTimeProperty> |
   <ApptByAttributeEndTimeProperty>                             (4)
<ApptByAttributePeopleProperty> ➔
   {<PreApptByAttributePeopleProperty>} <T_People>
   {<PostApptByAttributePeopleProperty>}                        (5)
<ApptByAttributeHead> ➔ NN                                      (6)
<PreApptByAttributePeopleProperty> ➔ .*
<T_UpdateAppt> ➔ <C_AddAttendee>
<C_AddAttendee> ➔
   <AddAttendeeCmd> {<AddAttendeeProperties>}
<AddAttendeeCmd> ➔ .*
<AddAttendeeProperties> ➔
   <AddAttendeeProperty> {<AddAttendeeProperties>}
<AddAttendeeProperty> ➔ <AddAttendeePeopleProperty> |
   <AddAttendeeExistingApptProperty>
<AddAttendeeExistingApptProperty> ➔
   {<PreAddAttendeeExistingApptProperty>} <T_ExistingAppt>
   {<PostAddAttendeeExistingApptProperty>
<AddAttendeePeopleProperty> ➔
   {<PreAddAttendeePeopletProperty>} <T_People>
   {<PostAddAttendeePeopletProperty>
<PreAddAttendeeExistingApptProperty ➔ .*
<PostAddAttendeeExistingApptProperty ➔ .*
<PreAddAttendeePeopleProperty ➔ .*
<PostAddAttendeePeopleProperty ➔ .*
```

Here an *entity*, like ApptByAttributes, consists of a head, optional (in braces) modifiers that appear in front of the head (e.g. "*Alex's* meeting"), and optional properties that follow the head (e.g. "meeting *with Alex*") (rule 2). Both modifiers and properties are defined recursively, so that they finally incorporate a sequence of different slots (rules 3-4). Each slot is bracketed by an optional preamble and postamble (rule 5). The heads, slot preambles and postambles are originally placeholders (.*). Some placeholders are specified with part-of-speech constraints --- e.g., head must be a NN (noun). For a *command* like AddAttendee, the template starts with a command part <AddAttendeeCmd>, followed by <AddAttendeeProperties>. The rest is very similar to that of the template rules for an entity. The template sets up the structural skeleton of a grammar. Hence the task of grammar learning becomes to learn the expressions for the pre-terminals like heads, commands, preambles, etc. The placeholders, without any learning, can match anything and result in ambiguities. When the learned grammar is used in our experiments, the placeholders were allowed to match any input string with a large penalty. The non-terminal <T_People> is application dependent and therefore will be provided by the developer (in the form of a name list in this example).

3.2 Annotation: Reducing the Search Space

The annotation reduces the search space for the rewriting rules for the pre-terminals: the annotated slots serve as the divider that localizes the learning space. For example, with the semantic annotation in Fig. 1 and the template CFG, our robust parser [3, 15] can obtain the partial parse in Fig. 2, where the pre-terminals in *italic* are place-holders that are not matched with any word in the sentence, because neither the template grammar nor the annotation provides sufficient information for the correct decision. The terminals in bold face are matched to the pre-terminals according to the template grammar or the annotation. For example, *Ed* and *Alex* are respectively attached to the two T_People positions due to the constraints from the annotation. *Meeting* is associated with ApptByAttributeHead because it is the only remaining NN found by POS tagger and the template grammar requires that the head be a NN. *Invite,* which appears in front of *Ed,* can match both AddAttendeeCmd and PreAddAttendeePeopleProperty. Since the latter is optional, it is matched against the former. The remaining words, *to, the,* and *with,* cannot be deterministically aligned to any pre-terminals by the parser. However, given the partial parse tree, they can only be aligned with those pre-terminals in italic. This effectively reduces the search space for possible alignment.

```
AddAttendee
  AddAttendeeCmd invite
  AddAttendeeProperties
    AddAttendeeProperty
      AddAttendeePeopleProperty
        T_People Ed
        PostAddAttendeePeopleProperty
    AddAttendeeProperties
      AddAttendeeProperty
        AddAttendeeExistingApptProperty
          PreAddAttendeeExistingApptProperty
          T_ExistingAppt
            C_ApptByAttribute
              ApptByAttributeHead meeting
              ApptByAttributeProperties
                ApptByAttributeProperty
                  ApptByAttributePeopleProperty
                    PreApptByAttributePeopleProperty
                    T_People Alex
```

Fig. 2. Parse tree obtained with the template grammar and the annotation for the sentence *"Invite Ed to the meeting with Alex"*.

3.3 Pre-terminal/Text Alignment

Syntactic clues are then used to align the remaining words in this example. Prepositions and determiners can only combine with the word behind them, hence "to the" cannot align with the pre-terminal *PostAddAttendeePeopleProperty*. This leaves *PreAddAttendeeExistingApptProperty* the only choice. For the same reason, *PreApptByAttributeStartTimeProperty* is the only pre-terminal that *with* has to be aligned with. Therefore we can induce the following rules:

PreAddAttendeeExistingApptProperty → to the

PreApptByAttributePeopleProperty.→ with

Sometimes syntactic clues are not enough to resolve all the ambiguities. In this case, the system prompts the developer for the right decision. We are working on an alignment model based on Expectation-Maximization to do this automatically.

4. EXPERIMENTAL RESULTS

We have applied this algorithm to MiPad, a multimodal personal digital assistant for personal information management [16]. For these experiments, we only used the calendar portion of the data. This training data contains ~650 spoken sentences, whereas the test data contains 500 sentences. Both training and test data were manually annotated against the schema.

We used both the manually authored grammar in [16] and the grammar learned with the described procedure to parse the sentences to the semantic representation using our robust parser [3, 15]. The manually authored grammar was created with about 2.5 man-months and contains about 3000 rules. From them, the definitions for common semantic classes (<Date>, <Time>, <Duration>, <OfficeNumber>, etc.), approximately 1400 rules, were stored into a library that was reused by the learned grammar.

We analyzed the topic ID and slot ID performance of the two grammars. Topic ID was measured by extracting the top-level semantic token (14 alternatives) from a parse tree. Differences in topics for this task are sometimes very subtle. For example, *"show meetings today"* and *"show the meeting today"* were labeled as *ShowCalendar* and *ShowAppt* respectively in the reference set. Slot ID was extracted by listing all the paths from the root to the pre-terminals in the semantic parse tree. Hence a topic ID error will cause all the slots in a parse tree to be incorrect. Slot ID is difficult due to the fact that some slot contents (like meeting subject) cannot be predicted and hence have to be modeled with wildcards. We compared the slots of a parse tree with those in the corresponding manual annotation and added the insertion, deletion and substitution error rates.

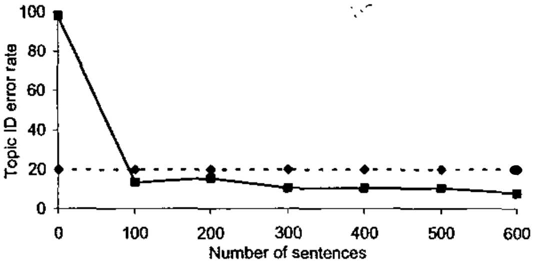

Fig. 3. Topic ID error rate vs. amount of annotated data for the learned grammar (solid line). The manual grammar is in dashed line.

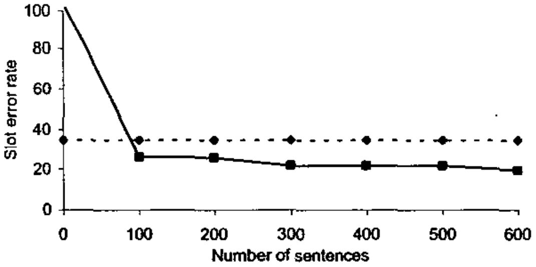

Fig. 4. Slot error rate (Ins+Del+Sub) vs. amount of data used for the learned grammar (solid line). The manual grammar is in dashed line.

Fig. 3 shows the Topic ID error rate of the learned grammar relative to the amount of annotated training data, as well as the performance of the manually authored grammar. With all the training data, the learned grammar reduces the error rate by 60% over the manually authored grammar. Moreover, the most significant error reduction was achieved with the first 100 annotated sentences. This is very meritorious since a typical developer is not likely to collect and annotate a large amount of data. Fig. 4 shows the slot ID error rate, which has a pattern very similar to Fig. 3.

0-7803-7343-X/02/$17.00 © 2002 IEEE

We used some other criteria to compute the slots error rates. The learned grammar consistently reduces the error rate by 40% ~ 60%. Due to the space limit, the performance curves are not included in this paper, but they are all similar to Fig. 4.

We also investigated the performance of the two grammars on speech directly. Fig. 5 plots the slot error rates when the input to the grammars is the top choice of a speech recognizer that uses various n-gram language models with corresponding different word error rates. The learned grammar consistently outperformed the manually author grammar even for speech input.

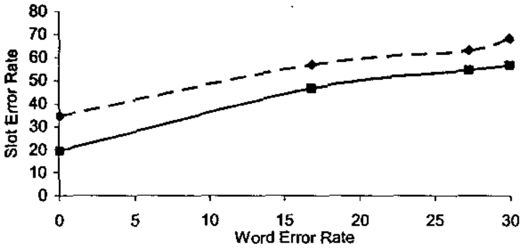

Fig. 5. Slot error rate vs. word error rate. The learned grammar is in solid line, the manual grammar in dashed line.

5. DISCUSSIONS AND SUMMARY

Our preliminary results indicate that the proposed machine aided grammar development framework is very promising. It achieves consistently better understanding accuracies with much less authoring effort than the manually authored grammar. We believe that the better performance is due to:

1. *Data driven learning*. Manually developed grammars tend to over-generalize the grammar for broader coverage. This generalization often does not cover any unseen data and it often results in ambiguities.

2. *The template grammar*. This formalism provides a paradigm for encoding semantic constraints and hence systematic structural generalization. We believe that the same paradigm may help human grammar developers too.

3. *The use of multiple information sources*. The grammar library makes it unnecessary to learn very low level concepts, and syntactic and annotation constraints segment the learning space so the learner can focus on the local points of interest.

The use of annotations in grammar development originated in Phoenix [1], where data were annotated against the semantic grammar. The annotations were used for grammar refinement by introducing new pre-terminal to terminal association rules. The benefit of our proposed approach is that we annotate against a schema that is independent of surface-structures. Because of this, we no longer need an initial semantic grammar, and the annotation task becomes much easier: For the ATIS grammar we'd only need to annotate text to the ~30 concepts instead of the 3000+ non-terminals. The cost of doing this is that the system has to be able to induce the CFG structures, which we have shown is viable with our proposed approach.

The MiPad data has only been used inside Microsoft. So far there is no comparison between our manually authored grammar and any other spoken language understanding technology, except for the comparative study that used the structured language model for information extraction [17]. To study the general applicability of our approach and provide the research community with more informative results, we intend to evaluate our approach with the ATIS data.

6. ACKNOWLEDGEMENTS

The slot ID evaluation criteria and the measuring tool were initially developed by M. Mahajan. We also like to thank C. Chelba, H. Hon, X. D. Huang, M. Mahajan, and K. Wang for their constructive suggestions and comments.

7. REFERENCES

[1] W. Ward, "Understanding Spontaneous Speech: the Phoenix System," ICASSP, Toronto, Canada, 1991.

[2] V. Zue et al, "JUPITER: A Telephone-Based Conversational Interface for Weather Information," *IEEE Transactions on Speech and Audio Processing*, 2000.

[3] Y. Wang, "Robust Spoken Language Understanding in MiPad," Eurospeech, Aalborg, Denmark, 2001.

[4] A. Waibel, "Interactive Translation of Conversational Speech," *Computer*, vol. 29, 1996.

[5] J. Glass and E. Weinstein, "SPEECHBUILDER: Facilitating Spoken Dialogue System Development," Eurospeech 2001, Aalborg, Denmark, 2001.

[6] M. Gavalda, "Growing Semantics Grammar." Ph.D Thesis, Carnegie Mellon University, 2000.

[7] K. S. Fu and T. L. Booth, "Grammatical inference: Introduction and survey, part 1," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 5, 1975.

[8] K. S. Fu and T. L. Booth, "Grammatical inference: Introduction and survey, part 2," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 5, 1975.

[9] R. C. Carrasco and J. Oncina, "Grammatical Inference and Applications," *LNAI 862*, Springer-Verlag, 1994.

[10] L. Miclet and C. d. l. Higuera, "Grammatical Inference: Learning Syntax from Sentences," *LNAI 1147*, Springer-Verlag, 1996.

[11] V. Honavar and G. Slutzki, "Grammatical Inference," LNAI *143*, Springer-Verlag, 1998.

[12] Y. Wang and A. Waibel, "Modeling with Structures in Statistical Machine Translation," COLING/ACL98, Montréal, Québec, Canada, 1998.

[13] A. Stolcke and S. M. Omohundro, "Inducing Probabilistic Grammars by Bayesian Model Merging," in *Proceedings of the Second International Colloquium on Grammatical Inference and Applications*. 1994.

[14] K. Wang, "A Plan-Based Dialog System with Probabilistic Inference," ICSLP, Beijing, China, 2000.

[15] Y. Wang, "A Robust Parser For Spoken Language Understanding," Eurospeech, Budapest, 1999.

[16] X. Huang et al, "MiPad: A Multimodel Interaction prototype," ICASSP, Salt Lake City, Utah, 2001.

[17] C. Chelba and M. Mahajan, "Information Extraction Using the Structured Language Model," EMNLP, Pittsburgh, 2001.