IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 1, pp. I-41- I-44, IEEE, Orlando, Florida, 2002

EVALUATION OF SPOKEN LANGUAGE GRAMMAR LEARNING IN THE ATIS DOMAIN

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

ABSTRACT

To facilitate the development of speech enabled applications and services, researchers have been working on a variety of smart tools. Recently we have introduced a schema-based context free grammar learning algorithm aiming at the development of real applications. In that paper we have described the algorithm and given some experimental results on the data of our in-house project. To study the general applicability of the algorithm as well as to provide the research community with more informative results, we have applied the algorithm to the well studied ATIS task and compared the performance of the learned grammar with one of the best performers in ATIS evaluations. The results show that the semi-automatically learned grammar achieves comparable performance to the manually authored grammar.

1. INTRODUCTION

While conversational systems have been developed in research labs for many years, speech-enabled conversational systems are still not mainstream in the real word. One of the problems, as we perceive it, is the requirement of intensive language engineering. For example, semantic-based robust understanding is a successful technology that has been widely used in research conversational systems [1-4]. However, 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.

To facilitate the development of speech enabled applications and services, researchers have been working on tools for rapid development of mixed-initiative systems [5, 6]. In [6], we introduced 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. The preliminary results indicate that the proposed machine aided grammar development framework is very promising: the technique not only significantly reduces the effort in grammar development, but also improves the understanding performance: the learned grammar consistently reduces the error rate by 40% ~ 60%.

However, the results in [6] were obtained from the experiments with our internal data in the domain of personal information management. No comparative studies have been conducted between our baseline system using manual grammar and other technologies. Therefore the improvement over the baseline in [6] may not be convincing. Furthermore, the preliminary success may just be an artifact of the fact that we have worked on the problem for a long period of time and the solution may simply be well suited to that particular problem.

To study the general applicability of the algorithm as well as to provide the research community with more informative results, we decided to apply our approach to a well studied problem in the public domain. The Airline Travel Information System (ATIS) [8] comes naturally as a choice since many labs have participated in ATIS evaluations. Another reason to use ATIS for evaluation is that we have the access to one of the best SLU performers in many years of the DARPA-sponsored ATIS evaluations, the CMU Phoenix/ATIS system.

In this paper, we will report some experimental results of our grammar authoring system on the ATIS data and compare it with the CMU system. The paper starts with a brief introduction of the algorithm, followed by a detailed description of the experimental setting and results, and ends with some discussion.

2. GRAMMAR LEARNING REVIEW

The learning algorithm starts with a semantic *schema* that defines the entity relations of a specific domain. It can be viewed as the specification for a language-enabled application. Schemas are language independent in the sense that they do not specify the linguistic expressions used to express the semantic entities. Schemas are manually authored by application developers. Because of language independency, it is much simpler to develop a schema than the linguistic context free grammars for spoken language understanding. For example, the ATIS schema can be printed out in 2 pages, while the CMU ATIS grammar requires 200+ pages. We found that an average developer with good understanding of the application that he is developing can easily author the semantic schema in half a day for a fairly complicated application in the PIM domain. Moreover, since semantic schemas play a critical role in dialog management [7], they have to be developed anyway in a multi-modal application; therefore it is not an extra burden for developers. The following is an example of concept definitions in a semantic schema:

```
<entity type="ExistingAppt" name="ApptByAttribute">
    <slot type="People"/>
</entity>
```

From the schema, a skeleton context free grammar can be automatically derived according to some templates. This skeleton CFG incorporates the semantic constraints specified in the schema. For example, the command in the previous schema example can derive the following CFG pieces:

```
<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>

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 post-amble (rule 5). The heads, slot preambles and post-ambles are originally placeholders (.*). Some placeholders could be specified with part-of-speech constraints --- e.g., head must be a NN (noun). A different template can be used to generate the CFG for a *command* semantic class. The templates set up the structural skeleton of a grammar. The placeholders, without any learning, can match anything. This makes the grammar greatly under-specified. A robust parser using this kind of grammars may result in too many ambiguous parses. The task of grammar learning, therefore, becomes to learn the expressions for these placeholders such that the grammar could be more specific and less ambiguous.

In addition to the automatically derived skeleton CFG, some domain-independent low level semantic entities, such as date, time, duration, postal address, currency, numbers, percentage, etc, can be written once and then shared by many applications. The grammars for these entities are manually authored and placed in a grammar library. Our grammar authoring tool allows users to associate a semantic concept in the schema with grammar rules in a grammar library. The grammar library may also contain application-dependent lexical rules. Normally these rules can be automatically obtained from the application-related database, for example, a list of airline names, aircraft codes, etc, in the airline travel domain.

To learn the linguistic expression for each of the placeholders in the skeleton grammar, our grammar learning tool takes advantage of the training data annotated against the schema. For example, the sentence "invite Ed to the meeting with Alex" is annotated against the schema in Fig. 1.

```
<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.

The annotations are used to reduce the search space for the grammar rules related to those placeholder pre-terminals. Our robust parser can find the parse of a sentence that satisfies the constraints specified in the semantic annotation of the sentence. In the parse, some input segments must match the CFG non-terminals that correspond to the marked semantic constituents in the annotation. These segments serve as the anchor points and divide the words-to-pre-terminal alignment space: the words that appear before/after an anchor point can only align to those pre-terminals that can legally appear before/after those anchor point non-terminals according to the skeleton CFG derived from the schema. This, together with some syntactic constraints described in [6], greatly reduces the search space and makes the learning plausible.

3. EXPERIMENTAL SETTING

To apply our technique to ATIS data, we need to first define the schema of the application, and then annotated the training sentences according to the schema. We did this with the help of the Phoenix/ATIS system.

3.1 Phoenix/ATIS System

The Phoenix spoken language understanding system was used by CMU in the ATIS evaluation [1]. It uses frames to represent semantic relations. A frame represents some basic type of action for the application. Slots in a frame represent the information that is relevant to the action. Slots in a frame are filled by matching patterns in an input string (sentence). The slots are filled independent of the order in which they appear in the frame. The patterns which fill slots are represented as Recursive Transition Networks (RTNs). Each slot has a grammar which specifies the patterns (strings of words) that can fill the slot. Since the patterns are compiled into RTNs, they can include calls to other networks. During parsing, the system uses slot-nets to match substrings in the input sentence. When a slot-net matches a substring, it is passed along for incorporation into frames. The system uses a beam search for frames. When a slot matches, it will extend all active frames that contain that slot. It will also activate any currently inactive frames that contain the slot. At the end of the sentence, the single best parse is returned from the beam.

3.2 Evaluation Criteria

The original DARPA-sponsored ATIS evaluation was conducted with a backend database. Sentences are first parsed by Phoenix with the ATIS RTNs. The analysis results were then translated into database queries to obtain the information requested by the user. The obtained database entries were compared with the target entries manually labeled for each sentence. A target entry is a pair of minimum and maximum information, which indicates the columns that have to be returned (minimum information) and the maximum columns that are allowed to be returned from the database. The minimum-maximum target evaluation mechanism penalizes the implementation that always returns all information regardless of what users have requested for.

Unfortunately, the translation from the analysis results to SQL queries depends on the grammar used in the analysis. Hence the translator for Phoenix/ATIS cannot be used by our system with learned grammar. The implementation of the translator is as hard as manual grammar development, if not harder. If possible, we would like to avoid implementing the translator which is not

essential in this study. Fortunately enough, we have the entire Phoenix system. This makes it possible to evaluate the performance with respect to a canonical semantic annotation instead of the database query results --- For Phoenix/ATIS, if the semantic schema is designed in such a way that it is very close to the CMU ATIS grammar, the parsing result can be automatically converted into the semantic annotations like the one in Fig. 1, and then compared with manual annotations to get the performance. For our robust parser/learned grammar, it directly generates the semantic markup of a sentence. The markups are then compared with the manual annotation. The number of insertions, deletions and substitutions of slots in the annotation were then collected for performance evaluation.

3.3 ATIS Schema

To facilitate the automatic conversion from Phoenix/ATIS parses to the semantic annotation, we have designed the schema to be as close to the Phoenix ATIS grammar as possible. This was achieved by looking at the frame definitions of the ATIS grammar for Phoenix. Strictly speaking, the slots in Phoenix/ATIS are not *semantic* slots. They are linguistic chunks that hold semantic slots together. For example, the slot "ARRIVE" for the frame "ShowFlight" actually contains multiple semantic slots like arrive_data, arrive_time and arrive_location. These semantic slots may also be included in other Phoenix slots, such as ARRIVE_DATE_RANGE, ARRIVE_LOC, etc.

To find the semantic slot that is related to a semantic class (frame), we followed the Phoenix/ATIS slot rules and extracted the semantic concepts like arrive_data, arrive_time. The final schema contains 8 top level semantic classes and 46 slots for those classes.

3.4 Grammar Library

Some application dependent CFG lexical rules are converted from the Phoenix/ATIS RTNs to non-terminal rules in our CFG grammar. The non-terminals include:

aircraft_name, airline_code, airline_name, airport_code, airport_name, cityname, class_type (e.g. business coach, etc), fare_basis_code, flight_number, meal_type (e.g. breakfast, lunch, dinner, etc), one_way (e.g. one way, round trip, etc), restriction_code, state, time, date, transport_type (e.g. taxi, bus, etc).

The grammar library also includes some domain independent entities such as date, time, duration, number, price, etc. This part of the library has been used in another domain for in [6].

3.5 Data

We used ATIS3 [8] training set A (sentences that can be interpreted without context, ~1600 sentences) to learn the grammar with our grammar authoring tool, and used the ATIS3 1993 test set A (~450 sentences) for testing.

3.6 Annotations

Both training and test data are analyzed with the Phoenix/ATIS system. The parsing results were converted to XML format. It is then converted to the schema annotation with an XSL stylesheet. The annotations are then manually checked to modify the parsing errors introduced by Phoenix/ATIS.

4. EXPERIMENTAL RESULTS

We studied the topic ID and slot ID performance of the two grammars. Topic ID performance was measured by comparing the parser-found frame/semantic class name of a sentence with the manually labeled one. In slot ID evaluation, slots were extracted by listing all the paths from the root to the pre-terminals in the semantic parse tree, and the resulting list was compared with that of the manual annotation. Hence a topic ID error will cause all the slots in a parse tree to be incorrect in the slot ID evaluation.

	Phoenix	Learned Grammar	Learned Grammar w/ extra data
Topic ID	5.52	11.03	5.06
Joint Topic & Slot ID	9.94	9.35	7.67

Table 1. Error rates for the ATIS task of Phoenix and our Robust Parser with the learned grammar both when using only the training data and when augmenting it with 9 sentences to cover topics missing in the training data. Topic ID refers to the top level command (show flights, show capacity, etc). Joint topic & slot ID refers to the full path from root to a leaf slots, i.e. it also includes the corresponding slots (Seattle, New York, etc).

Table 1 compares the error rates between the Phoenix/ATIS analysis and our robust parser [3] with the learned grammar (column 2, 3). It appears that Phoenix/ has much better performance in topic ID than our system trained with the ATIS training data. This is due to the fact that some topics that appeared in the test data has never occurred in the training data. For example, 4% of the test data are ShowCapacity sentences (e.g., "*what is the capacity of MD 10*"), and no single sentence in the training data covers that topic. While this may be less of a problem for a grammar authored by an expert with good knowledge of the application domain, it is impossible for a data driven system like ours to learn the grammar rules for that topic from nothing.

However, in the real scenario of grammar development, this problem rarely happens. The grammar developer would provide samples for all the semantic classes in the applications. Actually our learning tool is capable of warning developers that certain semantic/slots have never been covered by training samples and therefore asking developers to provide extra samples for them. To see how this helps, we ran another experiment where we augmented the ATIS training data with 3 sentences for each of the 3 topics that our system has prompted for more data:

ShowCapacity:
What is the capacity of the aircraft m eight zero
How many people can m eight zero hold
Tell me the capacity on m eight zero
ShowAirlineServeCity:
List the cities served by united
List the cities that united flies to
What cities does united serve
ExplainCode:
Which airline is A S
What is the abbreviation U S
Tell me about the m eighty aircraft

As shown in Table 1, augmenting training data this way significantly improves both topic ID and slot ID to the point that the combined system reduces the Topic ID and the joint topic&slot ID error rates are lower by 8% and 23% repectively over those of the Phoenix system. As in [6], we also investigated the effect of training data on the performance. Fig. 2 shows the Topic ID error rates of the learned grammar relative to the amount of annotated training data, with and without the extra sentences listed above, as well as the performance of the manually authored Phoenix grammar. Fig. 3 illustrates the slot performance of the same grammars. These two figures show the similar meritorious property observed in [6]: the most significant error reduction was achieved with the first 200 annotated sentences. Therefore the algorithm does not require developers to collect and annotate a large amount of training data.

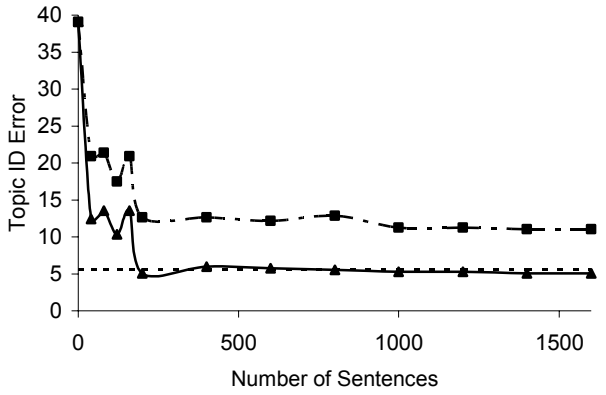

Fig. 2. Topic ID error rate vs. amount of annotated training data. The dashed horizontal line represents Phoenix/ATIS. The dashed curve represents the grammar trained with ATIS3 set A training data; the solid curve represents grammar trained with the extra sentences. Note that 200 sentences are generally sufficient.

5. DISCUSSIONS AND SUMMARY

The evaluation was somehow biased in favor of Phoenix/ATIS due to the following reasons:

1. The schema was derived from the Phoenix grammar. In other words, the semantic representation was designed to fit the Phoenix analysis grammar;
2. The test data annotation was derived from the Phoenix output. In case that ambiguous parses exist, only the one picked by the Phoenix parser is considered correct.

The grammar used by Phoenix for ATIS is very complicated. It consists of 3187 non-terminals and 13291 grammar rules. Our learned grammar is much smaller than that. There are 592 nonterminals (437 from grammar library and 155 from the skeleton grammar generated from the schema) and 3225 rules (2403 from the grammar library; 134 from the skeleton grammar and 688 learned from the annotated data.) The smaller grammar size not only makes the parser work faster, but also makes it easier to maintain.

Even with the bias against it, our robust parser achieved comparable performance to Phoenix/ATIS, without the intensive grammar authoring effort. The learned grammar is smaller and follows a common paradigm. This makes the grammar much easier to maintain.

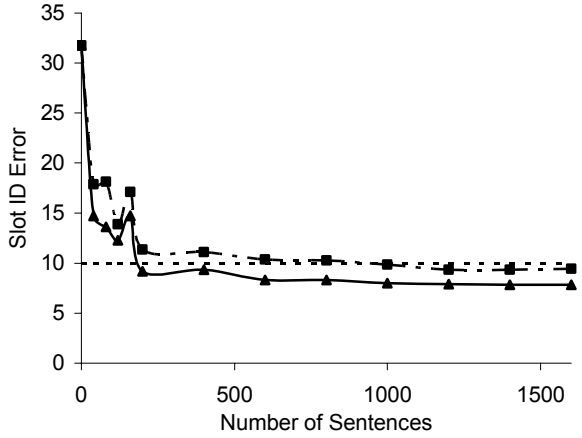

Fig. 3. Slot error rate (Ins+Del+Sub) vs. amount of annotated training data. The dashed horizontal line represents Phoenix/ATIS. The dashed curve represents the grammar trained with ATIS3 training data; the solid curve represents grammar trained with the extra sentences. Note that 200 sentences are generally sufficient.

6. ACKNOWLEDGEMENTS

The authors would like to thank Wayne Ward, Alex Rudnicky and the CMU speech group for their help with Phoenix/ATIS. 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 and et al, "JUPITER: A Telephone-Based Conversational Interface for Weather Information," *IEEE Transactions on Speech and Audio Processing*, vol. 8, 2000.

[3] Y.-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, Aalborg, Denmark, 2001.

[6] Y.-Y. Wang and A. Acero, "Grammar Learning for Spoken Language Understanding," IEEE workshop on Automatic Speech Recognition and Understanding, Madonna di Campiglio, Italy, 2001.

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

[8] D. Dahl and et. al., "Expanding the scope of the ATIS Task: the ATIS-3 Corpus," Human Language Technology Workshop, 1994.