9th European Conference on Speech Communication and Technology, pp. 1869-1872, ISCA, Lisbon, Portugal, 2005

SGStudio: Rapid Semantic Grammar Development for Spoken Language Understanding

Ye-Yi Wang, Alex Acero

Speech Technology Group, Microsoft Research
One Microsoft Way, Redmond, WA 98052, USA
{yeyiwang, alexac}@microsoft.com

Abstract

SGStudio (Semantic Grammar Studio) is a grammar authoring tool that facilitates the development of spoken dialog systems and speech enabled applications. It enables regular software developers with little speech/linguistic background to rapidly create quality semantic grammars for automatic speech recognition (ASR) and spoken language understanding (SLU). This paper introduces the framework of the tool as well as the component technologies, including knowledge assisted example-based grammar learning, grammar controls and configurable grammar structures. Experimental results show that SGStudio not only greatly increases the productivity, but also improves the quality of the grammars developed.

1. Introduction

While hundreds of spoken dialog systems have been deployed in many different sectors, it is still very costly and laborious to develop such systems. To facilitate the rapid development of spoken dialog system, it is important to identify the major barriers that developers in the speech processing industry are facing. [1] analyzed the "chasm" between SLU research and industrial applications, and listed the potential areas of improvement that the research community can provide:

1. There is little data for training in the design/development phrase. This prohibits machine-learning techniques being used in the initial system developement. Developers often have to manually author grammars. Tools for fast grammar handcrafting are very important. Other tools like those for content word normalization/speechification are also very desirable.
2. There is a huge amount of data available after deployment. It is extremely difficult to manually analyze the data in order to find the problems in the initial deployment. Tools for automatic or semi-automatic adaptation/learning/system tuning are very useful for improving the system's performance.

SGStudio is a tool aimed at the problems in these areas. While it focuses on the first problem, its compotent technolgy, such as knowledge-assisted example-based modeling, can also be applied to attack the second problem.

The following section presents the architecture of SGStudio. The remaining sections describe the component technologies. Section 3 discusses the knowledge-assited data-driven statistical modeling; section 4 introduces the grammar controls; and section 5 shows how the learning outcome can be customized to fit different application scenarios.

2. SGStudio Architechture

Figure 1 shows the architecture of SGStudio. At the center is a statistical model that adopts a pattern recognition approach to SLU. Given the word sequence W, the goal of SLU is to find the semantic representation of the meaning M that has the maximum *a posteriori* probability $\Pr(M|W)$:

$$\hat{M} = \arg\max_M \Pr(M|W) = \arg\max_M \Pr(W|M)\Pr(M)$$

In the equation, the semantic prior model $\Pr(M)$ assigns a probability to an underlying semantic structure (meaning) M. The lexicalization model $\Pr(W|M)$ assigns a probability to the surface sentence W conditioned on the semantic structure. *HMM/CFG composite model* is a specific statistical model under this framework, which alleviates the data sparseness problem by incorporating domain knowledge.

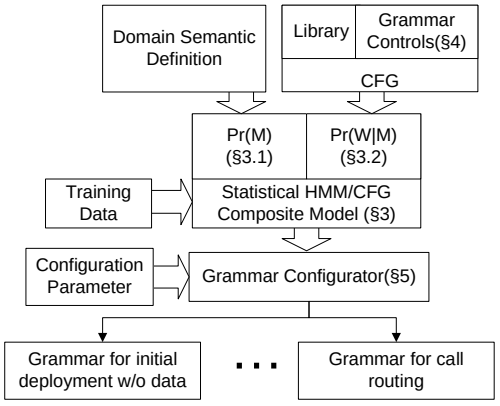

Figure 1: SGStudio architecture. At the center is the HMM/CFG composite model that incorporates domain knowledge, in the form of domain semantic definition and PCFG rules for domain-related concepts. The model can be configured with the configuration parameters to produce different grammars according to different application scenarios.

To create the domain-specific PCFG rules, *grammar controls* are introduced, which can automatically create quality ASR/SLU grammars from high level specifications.

Different developers often face different application scenarios, like the availability of the data, the complexity of the task, and the availability of human resources in system maintenance, etc. The *grammar configuration* module of SGStudio customizes the general HMM/CFG composite model to fit these scenarios.

3. Knowledge Assisted Example-based Statistical Modeling

SGStudio adopts HMM/CFG composite model for data-driven grammar learning. It integrates domain knowledge by setting the topology of the prior model, $\Pr(M)$, according to the domain semantics; and by using PCFG rules as part of the lexicalization model $\Pr(W|M)$.

The domain semantics define the semantic structure of an application with *Semantic frames*. Figure 2 shows a simplified example of two semantic frames in the ATIS domain.

<frame name="**ShowFlight**" type="*Void*">
  <slot name="DCity" type="*City*"/>
  <slot name="ACity" type="*City*"/>
</frame>
<frame name="**GroundTrans**" type="*Void*">
  <slot name="City" type="*City*"/>
  <slot name="Type" type="*TransType*"/>
</frame>

Figure 2: Simplified semantic frames in the ATIS domain. The type attribute of a slot restricts the type of its filler object. The "*Void*" type of the "**ShowFlight**" frame indicates that it is a top level command, a.k.a. a "task".

3.1. Semantic Prior Model

The HMM topology and the state transition probabilities comprise the semantic prior model. The topology is determined by the domain semantics defined by the frames and the transition probabilities can be estimated from the training data. Figure 3 shows the topology of the underlying states in the statistical model for the semantic frames in Figure 2.

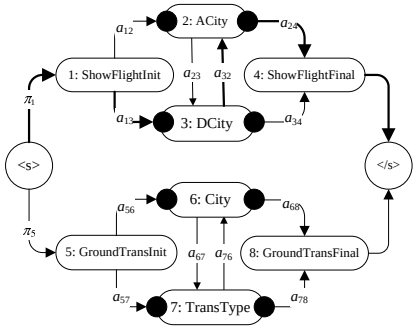

Figure 3: The HMM/CFG composite model's state topology, as determined by the semantic frames in Figure 1. State 1 and state 5 are also called the *precommands* for the **ShowFlight** and the **GrandTrans** frame, respectively. State 4 and state 8 are called the *postcommands*. States 2, 3, 6 and 7 represent slots. They are actually a three state sequence — each slot is bracketed by a *preamble* and a *postamble* (represented by the dots) that serve as the contextual clue for the slot's identity.

3.2. Lexicalization Model

The HMM/CFG composite model attempts to strike a balance between robustness and precision for spontaneous speech understanding. PCFG models, which impose relatively rigid restrictions, are used to model the slot fillers, which are more crucial for correct understanding and less subject to disfluencies because they are semantically coherent units. On the other hand, the sub-languages for precommands, postcommands, preambles and postambles, which glue different slot fillers together, are normally domain dependent, hard to pre-build a model for, and subject to more disfluencies. They also vary significantly across different speakers. The n-gram models are more lenient and robust to cover these sub-languages. Furthermore, the knowledge introduced by the PCFG sub-models greatly compensates the data sparseness problem — there is no need for the data to learn these ground level grammar structures. This is a major difference from other statistical SLU models like the one in [2]. Figure 4 shows a state alignment for the sentence "show flights departing from Boston to New York City" according to the "**ShowFlight**" network topology in Figure 3. The original preambles and postambles are replaced by the rounded rectangles. Since a slot is always bracketed by the preamble and postamble (though both of them may cover empty strings), the probabilities for the transitions between the preambles and the slot fillers and between the slot fillers and the postambles are always 1.0.

[3] introduced the learning algorithm for the HMM/CFG compsite model.

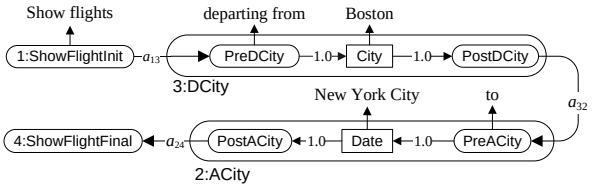

Figure 4: A zoomed-in view of a state alignment for the phrase "departing from Boston to New York City" according to the network topology in Figure 3. The output distribution of the rectangle states follows a PCFG, and that for the rounded rectangle states follows an n-gram model.

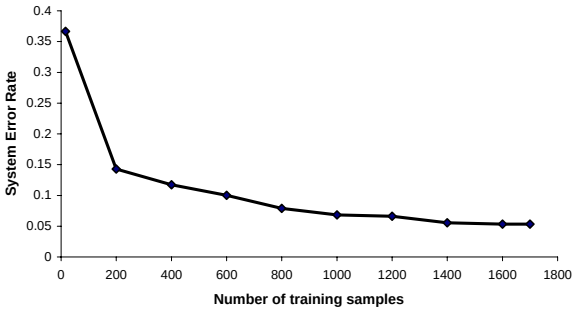

Figure 5: End-to-end system error rate on text input vs. amount of the training data

Figure 5 shows the effect of different amount of training data on the model's accuracy. With the standard ATIS evaluation, it took around half of the annotated 1993 ATIS3 training data sentences to achieve the nearly optimal accuracy that the system can achieve with more data. With a couple of samples per task, the system had a semantic accuracy at 60%, and the error rate drops significantly for the first couple hundreds of samples. Using all the 1993 training data, the error rate is 5.3%, which is comparable to the best system that used over 7300 ATIS2 and ATIS3 category A training sentences.

4. Grammar Controls

The HMM/CFG composite model exploits CFGs as the lexicalization models for slot fillers, which generally model a specific concept. If the concept is domain-independent, the grammar rules can be prebuilt in a grammar library. If it is domain-dependent like insurance policy numbers, users have to either use the closest generic CFGs in the library (e.g. generic alphanumeric sequence grammar), or build their own customerized grammar. The generic grammar has higher perplexity, which leads to higher error rate. The customized grammar is hard to author for regular developers. It is hard to anticipate the various expressions that refer to the same meaning; it is hard to normalize the various expressions with the semantic interpretation (SI) tags; and it is hard to optimize the grammar structure for best recognition performance. Grammar controls solve the problems by encapsulating the expert-level grammar implementation details in the controls. The controls can be built for the frequently used concepts. Users can customize the controls to their own needs with control parameters. The implementation details, including the anticipation of different expressions, the SI tags, the grammar structure optimization and the SRGS syntax, are all taken care of by the language technology experts who implement the controls.

Table 4 lists the basic grammar controls for some frequently used concepts.

Name	Description	Parameter	Example
ANC	Alphanum	RegExp	ANC(\d{3}-\d{4})
CAR	Card. num	Range/Set	CAR(1-31)
ORD	Ord. num	Range/Set	ORD(1, 2, 4, 8)
LST	Item list	String Items	LST(apple, pear)
LST	DB entries	Table Col.	LST(Svr:DB:City:name)

Table 1: Basic grammar controls. The ANC control generates grammars for alphanumeric concepts, such as insurance policy numbers, auto part numbers, etc. Its parameter is a regular expression that describes the pattern of the alphanumeric string. The CAR and ORD controls generate grammars for cardinal and ordinal numbers (non-negative integers), specified by either a parameter for the range of the numbers or a parameter that lists the numbers in a set. The LST control generates the grammar for a list of items. The parameter specifies either the items in the list or a column in a database table. The example in Table 4 shows the control that generates a City grammar from the "DB" database that resides on the server "Svr". Items are taken from the "name" column of the "City" table in the database to populate the grammar.

Besides the basic controls, we also introduced several control operations that yield more complicated grammars from the basic controls.

The concatenation operator $\otimes$ combines two operand rules sequentially to form a more complicated rule. "LST(April, June, September, November) $\otimes$ LST(the) $\otimes$ ORD(1-30)" generates the date grammar for the months with 30 days.

The paste operator $\oplus$ pair-wisely concatenates the entries in the operand rules. Assume that the table T of the database DB on the server S has a column for employees' first names and a column for their last names, then the paste operation "LST(S:DB:T:firstname) $\oplus$ LST(S:DB:T:lastname)" creates a grammar that correctly models all legitimate employee names.

The normalization operator $\odot$ associates, pair-wisely, an entry in the second operand rule as the normalized semantics of the corresponding entry in the first operand rule. Assume that the database table T in the previous example also has an employee ID column eid, "(LST(S:DB:T:firstname) $\oplus$ LST(S:DB:T:lastname)) $\odot$ LST(S:DB:T:eid)" results in a grammar that accepts an employee's name and returns his/her employee ID.

The left operand of the composition operator $\bullet$ must be an ANC control. "ANC$\bullet$LST(Svr:DB:City:cityname)" accepts spelling utterances like "S E A T T L E" or "S E A double T L E" and returns "Seattle" as the semantics for the two utterances.

Table 2 compares the character error rates and the semantic error rates between the library grammar and the customized grammar created by grammar controls in the recognition of social security numbers (SSN), license plate numbers (LPN), and Washington State driver license numbers (WADL). The generic grammar rules that are closest to the concepts were chosen from the grammar library of Microsoft Speech Application SDK in the experiment. The customized grammars have higher accuracies on all tasks. The WASL task has higher error rate. This is mainly due to the fact that WSDL has more (phonetically confusible) letters and many subject chose to pronounce the last names in WSDL instead of spelling them out.

220 Samples/Task		Library	Grammar Control
SSN	CER	7.5%	1.8%
	SER	22.3%	13.2%
LPN	CER	10.8%	4.9%
	SER	47.7%	22.7%
WSDL	CER	24.5%	23.6%
	SER	81.4%	63.2%

Table 2: Character error rate (CER) and semantic error rate (SER) for the library grammar and the grammar control created grammar.

5. Grammar Configurability

The HMM/CFG model was designed for mixed-initiative systems. It requires labeled training data even though the inclusion of the domain knowledge has significantly reduced the requirement. In many different application scenarios, simplified model topologies that require even less or no training data is more suitable. The simplified topology can be obtained with configuration paramaters in SGStudio.

There are two categories of parameters. The first includes one parameter that controls the overall model topology. The "backbone" parameter takes one of the two values, "Template_HMM" or "Domain_Ngram". In the case of "Template_HMM" setting, different paths are kept for different tasks, and the words that a state emits only depend on the history of words from the same state. On the other hand, the "Domain_Ngram" setting collapes all the paths for different tasks, and the the history of a word may also include the previous state or the words from the previous state, depending how the previous state is modeled. This is useful when the task identity is not important, for example, when the model is going to be used as a language model for speech recognition. The second category of parameters determine how the submodels are modeled. It includes four parameters for the preambles, postambles, precommands and postcommands. The parameters take one of the five

possible values. “None” means that the specific state should be omitted from the general model topology (e.g., slot postambles are often optional in English); “Wildcard” indicates that there is no specific language model for the state. A phone loop model should be used to accept any acoustic inputs; “PooledNgram” ties the model with the models of all the other “PooledNgram” states. The training data for all those states are pooled together. These three values eliminate or reduce the requirement for the training data. The value “Ngram” results in the standard composite model described in the previous section; and the “Rule” value lets the model use CFG rules instead of the n-gram models for the state.

As an example, we show how a concept spotting model can be created by SGStudio. When there is no data available for initial system development, developers often opt to write the core grammars and let the system handle the non-critical part of an utterance. For example, in a pizza ordering application, developers may want to just write the grammar for topping and size.However, users may say “I want to have a large pizza with mushroom and cheese.” The SLU component needs to spot the word “large” and “medium” for size and “mushroom” and “cheese” for toppings from the user’s utterances. A concept spotting model can fulfill this task, with a phone loop model that picks everything except for the key phrases (Figure 6).

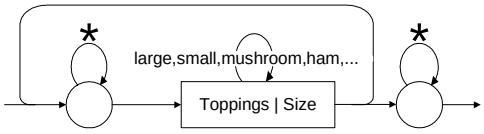

Figure 6: The concept spotting model: the key phrase model is bracketed by the wildcards (*) that can match anything a user may say. The task of recognition/understanding is to extract the key phrases from the user’s utterance.

Figure 7 shows the grammar configuration that generates such a model.

```
<GrammarConfiguration>
    <Preamble>Wildcard</Preamble>
    <Postamble>None</Postamble>
    <PreCommand>None</PreCommand>
    <PostCommand>Wildcard</PostCommand>
</ /GrammarConfiguration>
```

Figure 7: The grammar configuration that creates the wildcard model. The precommands and the postambles of the general topology in Figure 3 are omitted. All preambles are modeled with the wildcard so they collapsed into the left wildcard node in Figure 6, and all postcommands collapsed into the right wildcard node.

While the concept spotting model is robust and requires no training data, it often has high insertion error rate. Because the wildcard model is very flat, it tends to assign lower probability to the matched acoustic frames. Therefore the input acoustics are more likely to be matched with the key phrase model, which results in false positive errors. This model can be used to deploy an initial system, and more sophisticated systems can be built as the data are available after deployment.

The HMM/CFG composite model can also be configured for the call routing applications. If the command frames do not contain any slots, and the configuration omits all the preambles, postambles and postcommands and uses n-grams for precommands, we end up with the topology shown in Figure 8. For an input sentence W, the decoder picks the task that maximizes the *a posteriori* probability in equation 1, which is the n-gram classifier described in [4].

$$\text{Task} = \arg\max_{task_i} \Pr(task_i) \times \Pr(W|task_i) \qquad (1)$$

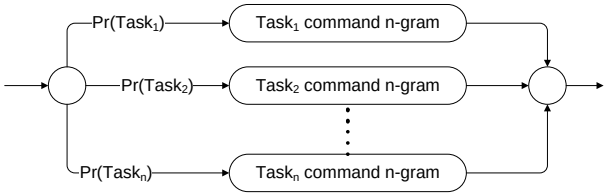

Figure 8: HMM/CFG model’s call-routing topology.

6. Conclusions

We have introduced SGStudio, a semantic grammar authoring tool that enables regular application developers to rapidly create ASR/SLU grammars. It exploits the technologies of the knowledge-assisted, example-based statistical modeling and grammar controls. The statistical HMM/CFG composite model integrates the domain knowledge in the data-driven gramamr learning framework. It significantly reduces the requirement for a large amount of traning data. Grammar controls, together with the control operations, are very powerful in generating various grammars for concepts that can be used in a system-initiatied dialog or as the filler of a slot in a mixed-initiative system. Coupling the two technologies together strikes a balance between the robustness and the constraints on overgeneralizations/ambiguities, achieves the accuracy better than or comparable to the best manually developed system, and greatly improves the grammar development productivity. We further described the configurability feature of SGStudio, which allows the creation of grammars customized to different application scenarios.

7. References

[1] R. Pieraccini, “Spoken language understanding, the research/industry chasm,” in *HLT/NAACL Workshop on Spoken Language Understanding for Conversational Systems*, Boston, 2004.

[2] S. Miller, R. Bobrow, R. Ingria, and R. Schwartz, “Hidden understanding models of natural language,” in *the 31st Annual Meeting of the Association for Computational Linguistics*, New Mexico State University, 1994.

[3] Y.-Y. Wang and A. Acero, “Concept acquisition in example-based grammar authoring,” in *ICASSP*, Hong Kong, China, 2003.

[4] Y.-Y. Wang, A. Acero, C. Chelba, B. Frey, and L. Wong, “Combination of statistical and rule-based approaches for spoken language understanding.” in *ICSLP*, Denver, Colorado, 2002.