Towards a new implementation approach for rapid development of text-based dialog systems

Zhibo Liu, Thomas Fang Zheng, Xiaojun Wu, and Mingxing Xu
Center for Speech Technology, National Laboratory for Information Science and Technology,
Tsinghua University, Beijing, China
{lzb, wuxj}@cst.cs.tsinghua.edu.cn, {fzheng, xumx}@tsinghua.edu.cn

Abstract

Dialogue Systems are now more and more used in real life. However, such systems are usually difficult for lay people to create. Some less complicated methods have been proposed, although with less powerful understanding capability. After introducing and comparing the approaches mentioned above, the grammar parsing part, named GrammarTool, of a natural language text interface Spoken Dialogue System (SDS) toolkit, named SDS Lite, is described in detail. Our system deals with Chinese dialogue and makes it very easy for lay people to learn and create their own SDS.

1. Introduction

Spoken Dialogue System (SDS) is now more and more used in real life, especially for information queries in restricted domains. However there are usually many different and unique ways of expression -- corresponding to unique grammar rules and keywords -- that frequently appear in such areas. Thus in order to develop an SDS, it's very important to extract domain specific grammar rules and keywords in advance (assuming a knowledge-based system) or have a large corpus library (assuming a statistics-based system). A knowledge-based approach is assumed in this paper because of the difficulty of collecting corpora for new domains.

There are already several mature and capable SDSs, created by CMU, MIT and other universities and organizations(Seneff, 1992; Tomko, Toth, Sanders, Rudnicky & Rosenfeld, 2005; Ward, 1990). In general, their systems process English while ours, named d-Ear SDS Lite (with a grammar generation tool named GrammarTool), processes Chinese.

Because of the broad utility of SDS, we consider it very important that lay people be able to develop their own systems. In general, such systems must be more constrained than a system developed by experts. Our system is one such system, whose goal is simple: develop a useful SDS toolkit that can be used by non-experts who know little about programming, natural language understanding or speech processing.

Our system is based on a subset of Enhanced Context Free Grammar, or ECFG (Yan, 2002). Of the five rule-types in this system (the strict rule, jumping rule, unordered rule, long-distance rule, and crossing rule types), GrammarTool currently only supports the jumping rule type, which is the most often used type among the five. Other rule types such as the unordered rule type will be taken into consideration in future work.

2. Related work

Several similar toolkits have been developed for a similar purpose. Two of them, Phoenix and the Application Generation Toolkit (Toth, Harris, Sanders, Shriver & Rosenfeld, 2002), are introduced below.

The Phoenix parser can be used to develop simple but robust SDS applications. The basic semantic unit is a frame, and Phoenix parses every input sentence into a set of frames. The target developer must manually define the frames and grammar rules used for each application.

The Application Generation Toolkit is a module of the Universal Speech Interface (USI) project (Rosenfeld, Zhu, Shriver, Toth, Lenzo and Black, 2000; Rosenfeld, Olsen and Rudnicky, 2000). It allows lay people who do not know a lot about programming to create and use fully capable SDS applications with a specific chosen database in a short time, as depicted in Figure 1(Toth et al., 2002).

Proceedings of the 11th Australian International Conference on Speech Science & Technology, ed. Paul Warren & Catherine I. Watson. ISBN 0 9581946 2 9
University of Auckland, New Zealand. December 6-8, 2006.

Accepted after full paper review

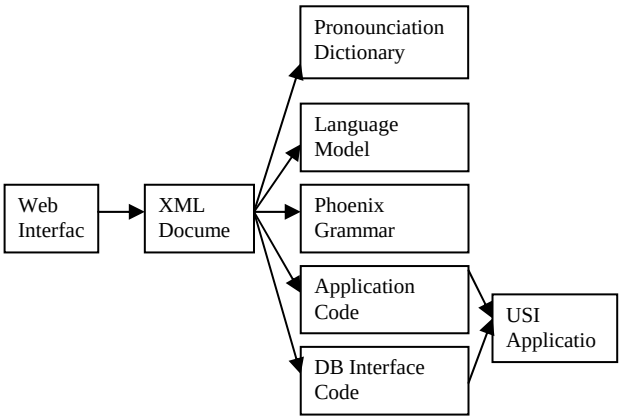

Figure 1. Application generation process

The Application Generation Toolkit makes it much easier for people to develop their own SDS. However, the target developers can just input some column names and other phrases on the webpage and the system only supports two input patterns. So in order to improve the usability the power of this toolkit is limited.

GrammarTool provides a "middle-of-the-road" solution: our toolkit is less complicated for target developers to learn and use than Phoenix, while at the same time more powerful (and thus slightly more complicated) than the Application Generation Toolkit.

3. General structure of SDS Lite

SDS Lite is composed of a toolkit and a runtime platform. Developers can use the toolkit's two tools, KeywordTool and GrammarTool, to create the domain-specific configuration files used by the runtime platform.

Before explaining KeywordTool and GrammarTool, three different types of keyword in the whole SDS system should be introduced firstly:

(1) *Predefined Keyword*: these are the commonly used words, phrases and their synonyms, like "你好"(hello), "我想问一下"(I want ask). They have been all collected and stored in a text file, which has 134 classes of predefined keywords and around 500 keywords.

(2) *User-defined Keyword*: these are the core concepts in the domain and should be defined in KeywordTool by user.

(3) *Extracted Keyword*: These are the keywords extracted from training sentences by GrammarTool and will be stored to be used in the specific domain.

KeywordTool is used to manage domain specific keywords and keyword classes. Take the query of NBA information as an example; in this domain, keywords might include player and team names. Two keyword classes ("player" and "team") could be added. KeywordTool supports multiple synonyms for a single keyword entry.

GrammarTool is used to customize the domain grammar, and will be described in more detail in the next section.

After finishing domain customization with KeywordTool and GrammarTool, the generated configure files can be directly used by the runtime platform.

4. Introduction to GrammarTool

We define a "topic" as follows:

A topic is a named set of possible input patterns (or "pseudo-sentences") such that we expect the system to reply with the same answer given the same pattern fillers. For example:

[player_stats]
我想了解一下<player>的统计资料
请说明一下<player>的统计结果
<player>的统计情况有吗

These three Chinese sentence patterns express a similar meaning (all of them ask for statistical information for an NBA player), and they form a topic named "player_stats". For each topic, grammar rules and additional "helper" keywords are extracted.

GrammarTool will process all the category names and sentences and then extract and save domain specific grammar rules and keywords. After this training process, the d-Ear SDS Lite parser can analyze and understand most of the queries in this domain.

When the system is deployed, user queries will be parsed and identified as belonging to a specific topic. Slot contents for that topic will also be determined. Answer generation is then performed based on this information (response templates must currently be manually written for each topic).

5. Algorithm design

The main function of GrammarTool is to extract semantic rules from example sentences.

In GrammarTool, the "semantics" of a sentence contains:

(1) A topic in the domain

(2) Slots and their contents. The slots represent the keyword classes while the content of the slots are the normalized keywords themselves.

Before extracting grammar rules, GrammarTool will do the following several steps of preprocessing at first.

Sentences in the same category will be placed in a group, and then all the words that appear in the predefined keyword list will be replaced by the name of the particular predefined keyword class. For example, if the predefined keyword file includes the following keyword class:

Proceedings of the 11th Australian International Conference on Speech Science & Technology, ed. Paul Warren & Catherine I. Watson. ISBN 0 9581946 2 9
University of Auckland, New Zealand. December 6-8, 2006.

Accepted after full paper review

[these]
这些

Then any sentence containing the word “这些” will have the word replaced by “<these>”.

The predefined keywords include some words with little substantial semantic meaning, for example “我想查一下” (“I’m looking for…”) and question suffixes like “吗” and “呢”. If a training sentence contains the words like these, GrammarTool will delete such words to obtain a new training sentence that can be added the training set to enlarge the coverage.

GrammarTool will also compare sentences in a topic to find words that might be considered optional. For example, if the topic is about flight query and contains the following two training sentences:

从<city_from>到<city_to>
<city_from>到<city_to>

Then GrammarTool will consider “从” as optional, and if the topic also contains:

从<city_from>飞往<city_to>的航班

GrammarTool will generate the following new training sentence and add it into this topic automatically:

<city_from>飞往<city_to>的航班

If there're multi optional keywords in one sentence, according to the algorithm description above, several grammar rules would be generated, which could cause over generation of grammar rules to some extent. Therefore it should be very cautious when judging whether a keywords is optional. At present actually a keyword would be considered as optional only when the structures of two sentences are nearly the same while only this component is different.

After these preprocessing steps, sentences with the same structure will be put into a single subgroup, which will then generate a single grammar rule. For two sentences to have the “same structure” means that they contain exactly the same user defined and predefined keywords, in the same order and same position, for example:

xxx <keyword A> xxx <keyword B> xxx

where “xxx” represents one or more words, which will be extracted keywords. For example, suppose there are the following two sentences in the same topic:

Sentence 1: 我想问一下<actor>的最新作品
Sentence 2: 我想知道一下<actor>的新片

Then suppose that there are the following predefined keywords:

[wen_yi_xia]	[de]
我想问一下	的
我想知道一下	

After the predefined keywords replacement, the sentence would be:

<wen_yi_xia> <actor> <de> 最新作品
<wen_yi_xia> <actor> <de> 新片

Obviously, these 2 sentences have the same structure and will be grouped into the same subgroup. GrammarTool will then generate a new keyword class:

[categoryM_keyword_N]
最新作品
新片
where M and N are non-negative integers.

Finally, this subgroup will generate one grammar rule:

<wen_yi_xia> <actor><de> < categoryM_keyword_N>

6. Tests and results

Our testing objective was to obtain preliminary results on the effectiveness of our Grammar generation method. Here are our steps to test.

First we’ll determine a domain and pick some sentences in this domain as data set.

Second the domain specific keywords will be collected from those sentences with KeywordTool.

Third some sentences in the data set will be selected as training sentences, put into GrammarTool and others as test sentences.

Using the generated configuration files, we deployed the system and used the test set to determine which sentences could be correctly identified as belonging to the topic.

Three domains with corresponding sentences were selected for training and testing:

NUM_AGE, with one category [current_age] and 30 sentences, asking the age of a person;

TIME_DAY, with one category [which_day] and 30 sentences, asking the date; and

NUM_SPEED, with one category [speed] and 50 sentences, asking the speed of a person or thing.

For each domain, we used three methods for picking the training sentences:

(1) Using 5 sentences as a training set and leaving the other sentences as the test set.

Proceedings of the 11th Australian International Conference on Speech Science & Technology, ed. Paul Warren & Catherine I. Watson. ISBN 0 9581946 2 9
University of Auckland, New Zealand. December 6-8, 2006.

Accepted after full paper review

(2) Using 10 sentences as a training set and leaving the other sentences as the test set.

(3) Using half of the sentences as a training set and leaving the other half as the test set.

When choosing n number of sentences from the data set, we used two strategies. Take the domain NUM_AGE, which has 30 sentences (5 of them will be chosen), as an example:

(1) Ordered Method. We took the sentences with the index of $5i + 1$, $5i + 2$, $5i + 3$, $5i + 4$, $5i + 5$ ($0 \le i \le 9$) as a training set and regarded the other 25 sentences as the test set. So there were 6 results.

(2) Artificial Method. Manually choose 5 "representative" sentences to cover the other sentences as much as possible. One result will be given.

We believe the "Artificial Method" is a fairly realistic picture of how GrammarTool will be actually used, because when entering the training sentences, the developer will tend to try to think of different ways of expression, while in our data set sentences with a very similar structure were close to each other and thus were chosen together into the training set using the "Ordered Method". The results of our tests are as follows:

Table 1: Test results for NUM_AGE

NUM_AGE		No. of sentences in training set		
		5	10	Half
Order Method	Maximum	72.0%	100.0%	100.0%
	Minimum	4.0%	50.0%	73.3%
	Average	32.0%	73.3%	86.7%
Artificial Method		60.0%	80.0%	100.0%

Table 2: test results for TIME_DAY

TIME_DAY		No. of sentences in training set		
		5	10	Half
Order Method	Maximum	40.0%	80.0%	73.3%
	Minimum	12.0%	25.0%	46.7%
	Average	28.7%	45.0%	60.0%
Artificial Method		40.0%	44.0%	85.0%

Table 3: test results for NUM_SPEED

NUM_SPEED		No. of sentences in training set		
		5	10	Half
Order Method	Maximum	71.1%	60.0%	92.0%
	Minimum	2.2%	47.5%	56.0%
	Average	34.7%	54.5%	74.0%
Artificial Method		55.6%	75.6%	90.0%

From the results, it can be seen that the "Artificial Method" outperforms the "Ordered Method" in nearly every situation, which confirms what was inferred before the test. Meanwhile, the results would get better along with the enlargement of training set. Furthermore, developers need only prepare 10-20 sentences for each topic in order to get an SDS with good quality, which we consider an acceptable task for the average person.

7. Conclusion and future work

In general, GrammarTool has the following unique characteristics:

(1) It is appropriate for system configuration by non-experts after short training.

(2) Only a small number of training sentences is required to obtain broad coverage in the domain.

(3) It is optimized for Chinese, especially accounting for the loose grammar structure.

There are several areas of improvement we are considering for GrammarTool, including:

(1) Support for the unordered rule type of ECFG.

(2) Support for simple multi-turn dialogue.

8. Acknowledgements

Our test data was provided by the Information Retrieval Laboratory of Harbin Institute of Technology.

9. References

Rosenfeld, R., Olsen, D. & Rudnicky A. (2000). A universal human-machine speech interface. *Tech. Rep. CMU-CS-00-114*. School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Rosenfeld, R., Zhu, X.J., Shriver, S., Toth, A., Lenzo, K. & Black A. W. (2000). Towards a universal speech interface. *Proceedings of International Conference on Speech Language Processing '00*.

Seneff, S. (1992): TINA a natural language system for spoken language applications. *Computational Linguistics*, 18(1), 61-86.

Tomko, S., Harris, T., Toth, A., Sanders, J., Rudnicky, A. & Rosenfeld R. (2005). Towards efficient human machine speech communication: The speech graffiti project. *ACM Transactions on Speech and Language Processing*, 2(1).

Toth, A., Harris, T., Sanders, J., Shriver, S. & Rosenfeld R. (2002). Towards every-citizen's speech interface: An application generator for speech interfaces to databases. *Proceedings of International Conference on Speech Language Processing* 1497–1500.

Ward, W. (1990). The CMU air travel information service: Understanding spontaneous speech. *Proceedings of the DARPA Speech and Natural Language Workshop*.

Yan, P.J. (2002). The study of natural language understanding in dialogue systems: [Thesis]. Beijing: Department of computer science and technology, Tsinghua University.