Rapid Development Process of Spoken Dialogue Systems using Collaboratively Constructed Semantic Resources

Masahiro Araki
Department of Information Science
Kyoto Institute of Technology
Matsugasaki, Sakyo-ku, Kyoto 606-8585, Japan
araki@kit.ac.jp

Abstract

We herein propose a method for the rapid development of a spoken dialogue system based on collaboratively constructed semantic resources and compare the proposed method with a conventional method that is based on a relational database. Previous development frameworks of spoken dialogue systems, which presuppose a relational database management system as a background application, require complex data definition, such as making entries in a task-dependent language dictionary, templates of semantic frames, and conversion rules from user utterances to the query language of the database. We demonstrate that a semantic web oriented approach based on collaboratively constructed semantic resources significantly reduces troublesome rule descriptions and complex configurations in the rapid development process of spoken dialogue systems.

1 Introduction

There has been continuing interest in the development methodology of spoken dialogue systems (SDS). In recent years, statistical methods, such as Williams et al. (2007) and Hori et al. (2009), have attracted a great deal of attention as a data-driven (i.e., corpus-driven) approach, which can reduce the troublesome manual coding of dialogue management rules. Statistical methods can also be applied to other components of SDS, such as semi-automatic construction of semantic interpreters and response generators. However the overall SDS development process still requires some hand coding, for example to establish the connection to the underlying application.

Another data-driven approach was designed to provide all of the SDS components with the goal of rapidly constructing the entire system (Kogure et al., 2001; Heinroth et al., 2009). This approach starts from a data model definition (and so can be regarded as a data-modeling driven approach) and adds rules and templates, which are used as task-dependent knowledge in an SDS. As a data model definition, Kogure et al. (2001) used a relational database (RDB) schema and Heinroth et al. (2009) used OWL, which is an ontology definition language in semantic web applications. Although these data-modeling schemata are familiar to developers of web applications, additional definition of rules and templates needed for an SDS is troublesome for ordinary web developers because such SDS-related rules require specialized knowledge of linguistics and speech application development.

We herein propose a new data-modeling driven approach for rapid development of SDS that is based on collaboratively constructed semantic resources (CSRs). We present an automatic generation mechanism of code and data for a simple SDS. In addition, we compare the proposed approach with an ordinary data-modeling driven approach that is based on a RDB. By using CSRs and the Rails framework of web application development, the troublesome definitions of rules and templates for SDS can be reduced significantly.

Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL), pages 70–73,
Seoul, South Korea, 5-6 July 2012.

The remainder of the present paper is organized as follows. Section 2 describes the proposed approach to a data-modeling driven development process for SDS based on CSRs. Section 3 compares the proposed approach with the previous RDB-based approach. In Section 4, the paper concludes with a discussion of future research.

2 Data-modeling driven approach based on CSRs

In this section, we explain our previous data-modeling driven approach and describe additional new functionality based on CSRs.

2.1 Object-oriented SDS development framework

We previously proposed a data-modeling driven framework for rapid prototyping of SDS (Araki et al., 2011). This includes a class library that is based on the class hierarchy and the attribute definitions of an existing semantic web ontology, i.e., Schema.org[1]. This class library is used as a base class of an application-specific class definition. An example class definition is shown in Figure 1.

```
@DBSearch
@SystemInitiative
class MyBook extends Book {
  int ranking
  static constraints = {
    name(onsearch:"like")
    author(onsearch:"like")
    publisher()
    ranking(number:true)
  }
}
```

Figure 1: Example of class definition extending existing class library.

In this example, the *MyBook* class inherits all of the attributes of the *Book* class of Schema.org in the same manner as object-oriented programming languages. The developer can limit the attributes that are used in the target application by listing them in the constraints section. On the other hand, the developer can add additional attributes (in this class, *ranking* attributes as the type of *integer*) in the definition of the class.

The task type and dialogue initiative type are indicated as annotations at the beginning of the class definition. In this example, the task type is DB search and the initiative type is user initiative. This information is used in generating the controller code and view code of the target SDS.

Using Grails[2], which is a Rails web application framework, the proposed framework generates the dialogue controller code of the indicated task type and the view code, which have speech interaction capability on the HTML5 code from this class definition. The overall concept of the object-oriented framework is shown in Figure 2.

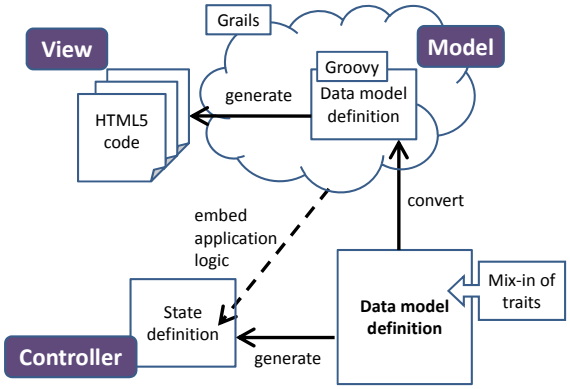

Figure 2: Overview of the object-oriented SDS development framework.

2.2 Usage of CSRs

The disadvantage of our previous framework, described in the previous subsection, is the high dependence on the dictation performance of the speech recognition component. The automatically generated HTML5 code invokes dictation API, irrespective of the state of the dialogue and initiative type. In order to improve speech recognition accuracy, grammar rules (in system initiative dialogue) and/or the use of a task/domain-dependent language model (LM) (in mixed/user initiative dialogue) are necessary. In our previous framework, the developer had to prepare these ASR-related components using language resources, which are beyond the proposed data-driven framework.

In order to overcome this defect, we add the Freebase[3] class library, which is based on large-scale CSRs, because Freebase already includes the

[1] http://schema.org/

[2] http://grails.org/

[3] http://www.freebase.com/

contents of the data. These contents and a large-scale web corpus facilitate the construction of grammar rules and a LM that is specific to the target task/domain. For example, the Film class of Freebase has more than 191 thousand entries (as of May 2012). These real data can be used as resources to improve SDS accuracy.

In system initiative type dialogue, the contents of each attribute can construct word entries of the grammar rule for each attribute slot. For example, the grammar rule for the user's response to "Which genre of movie are you searching for?" can be constructed from the contents of the genres attribute of the Film class. We implemented a generator of the set of content words specified in the data model definition from the data of Freebase. The generator is embedded as one component of the proposed rapid prototyping system.

In the mixed/user initiative type tasks, since content words and functional words make up the user's utterance, we need a LM for speech recognition and a semantic frame extractor for the construction of semantic data storage queries. We designed and implemented a LM generator and a semantic frame extractor using a functional expression dictionary that corresponds to the attributes of Freebase (Araki, submitted). An example entry of the function expression dictionary is shown in Figure 3 and the flow of the LM generation is shown in Figure 4.

item	value
property	fb:film.performance.actor
phrase pattern	X "ga de te iru" Y
constraints	X rdf:type "/film/actor"
partial graph	Y fb:film.performance.actor X

Figure 3: An entry of function expression dictionary.

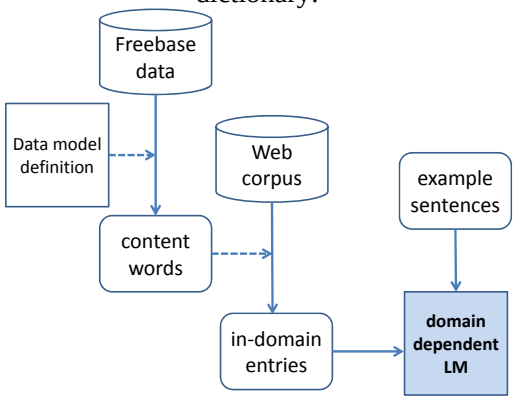

Figure 4: Construction process of LM.

3 Comparison with the RDB-based approach

3.1 Overview of the RDB-based method

As an example of the RDB-based SDS prototyping method, we review the method described in Kogure et al. (2001) (see Figure 5).

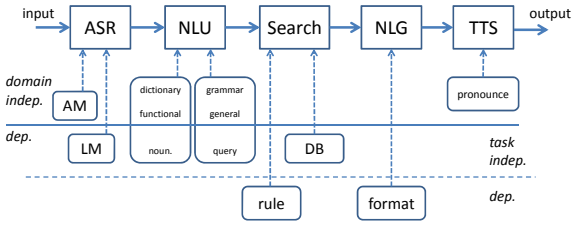

Figure 5: Modules and knowledge of the RDB-based method.

They examined the domain dependency and task dependency of the knowledge that drives SDS. Domain/task-independent knowledge, such as an acoustic model, a general function word dictionary, and a pronunciation dictionary, are prepared in advance for all of the systems. Both domain-dependent/task-independent knowledge, such as the language model, the noun/verb dictionary, and the database schema, and domain/task-dependent knowledge, such as the rule of query generation obtained from the results of semantic analysis and format for output, must be specified by the developer. If the developer wants to change a task within the same domain, the developer can reuse domain-dependent/task-independent knowledge (everything above the dotted line in Figure 4) and must specify task-dependent knowledge (everything below the dotted line in Figure 4).

3.2 Comparison of the data-modeling stage

In the data modeling of the RDB-based method, the developer must specify field names (e.g., title, year), their corresponding data types (e.g., string, integer), and the labels of the fields (i.e., the labels for the language used in the SDS), as in the usual web application with RDB. Since the data model definitions differ from one another, it is difficult to integrate similar systems even if these systems deal with the same domain.

In the CSRs-based approach, the data-modeling process involves selecting necessary attributes of the inherited class and, if needed, adding fields for

additional domain/task-specific information. The data type has already been set in the existing data schema, and language-dependent label information can be acquired by the value of *rdfs:label*, where the value of the *lang* attribute is the target language.

3.3 Comparison of code generation stage

In the RDB-based method, the developer must specify the noun and verb dictionary, grammar for parsing, and rules for query generation. In addition, the RDB-based approach must either stick to a fixed dialogue pattern for DB search or make the developer write dialogue management rules.

By combining the CSRs-based approach with the Rails framework, the task dependent dictionary is automatically generated from the data and grammar rules are easily constructed with the functional expression entries of properties. Also in this approach, typical dialogue management patterns are already prepared and can be specified as annotations. For the sake of this setting, all of the basic codes for SDS are automatically generated from the data model definition.

3.4 Comparison of functionality

In the RDB-based method, the developer must make a domain/task dependent LM using language resources outside of the development process. However, in general, it is difficult to acquire a domain/task-dependent corpus. In addition, although the RDB-based method is designed to be robust with respect to the task modification, this method is not robust with respect to porting to different languages. Language specific code tends to be embedded in every component of an SDS.

In the CSRs-based approach, the domain/task-dependent LM is automatically generated, as described in Subsection 2.2. For the sake of this data-modeling driven method and native multilinguality of CSRs, the developer can easily implement multilingual SDS (Araki et al., 2012). Multilingual contents are already prepared in Freebase (although English resources are dominant) and a multilingual web speech API is already implemented, e.g., in the Google Chrome browser, the developer can implement a prototype of other language SDS by dictation. If the developer wants to use domain/task-dependent LMs, he/she must prepare example sentences for the target domain/task in the target language.

4 Conclusions and future research

We have proposed a method for rapid development of a spoken dialogue system based on CSRs and have compared the proposed method with the conventional method, which is based on RDB.

In the current implementation, our system cannot handle the problem of the variation of the named entity which is dealt with by e.g. Hillard et al. (2011). We are planning to examine the extensibility of the proposed framework by combining such refinement methods.

Acknowledgments

The present research was supported in part by the Ministry of Education, Science, Sports, and Culture through a Grant-in-Aid for Scientific Research (C), 22500153, 2010.

References

Masahiro Araki and Yuko Mizukami. 2011. Development of a Data-driven Framework for Multimodal Interactive Systems. In Proc. of IWSDS 2011, 91-101.

Masahiro Araki. submitted. An Automatic Construction Method of Spoken Query Understanding Component from Data Model Definition.

Masahiro Araki and Daisuke Takegoshi. 2012. A Rapid Development Framework for Multilingual Spoken Dialogue Systems. In Proc. of COMPSAC 2012.

Tobias Heinroth, Dan Denich and Gregor Bertrand. 2009. Ontology-based Spoken Dialogue Modeling. In Proc. of the IWSDS 2009.

Dustin Hillard, Asli Çelikyilmaz, Dilek Z. Hakkani-Tür, and Gökhan Tür. 2011. Learning Weighted Entity Lists from Web Click Logs for Spoken Language Understanding. In Proc. of Interspeech 2011, 705-708.

Chiori Hori, Kiyonori Ohtake, Teruhisa Misu,. Hideki Kashioka and Satoshi Nakamura. 2009. Statistical Dialog Management Applied to WFST-based Dialog Systems. In Proc. of ICASSP 2009, 4793-4796.

Satoru Kogure and Seiichi Nakagawa. 2001. A Development Tool for Spoken Dialogue Systems and Its Evaluation. In Proc. of TSD2001, 373-380.

Jason D. Williams and Steve Young. 2007. Partially Observable Markov Decision Processes for Spoken Dialog Systems. Computer Speech and Language, 21(2), 393-422.