WFSTDM Builder – Network-based Spoken Dialogue System Builder for Easy Prototyping

Etsuo Mizukami and Chiori Hori

Abstract
This paper introduces a network-based spoken dialog system development tool kit: WFSTDM Builder developed by NICT. WFSTDM Builder provides functions to share and edit SLU and scenario so that developers can create a wfst-based spoken dialogue system instantly with this tool. One can test the scenario by accessing to the servers connected such as ASR, TTS and WFSTDM server via not only the tool’s web viewer and also via a dialog client application provided to realize spoken dialog. WFSTDM is an expandable and adaptable dialogue management platform. It combines various WFSTs and enables us to develop new dialogue management WFSTs necessary for rapid prototyping of spoken dialogue systems. In this paper, we illustrate the outline of WFSTDM Builder and its usage.

1 Introduction

There are many platforms which support rapid prototyping of dialogue systems and control dialogues while handling speech or multimodal interfaces. Some of them employ textual languages such as VoiceXML [1] to describe dialogue scenarios. Some systems accept table or automaton representation that can be designed with a graphical editor [2]. Other systems enable developers to use existing programming languages [3]. To construct an expandable and adaptable dialogue system which handles multiple tasks, we have proposed an efficient approach to manage the dialogue system using the weighted finite state transducer (WFST) [4].

Etsuo Mizukami
National Institute of Communications and Information Technology, 3-5, Hikaridai, Seikacho, 619-0289, Kyoto, e-mail: etsuo.mizukami@nict.go.jp

Chiori Hori
National Institute of Communications and Information Technology, 3-5, Hikaridai, Seikacho, 619-0289, Kyoto, e-mail: chiori.hori@nict.go.jp
Although WFSTs are mainly used in speech and language processing [5], we applied them for dialogue management where input symbols of the WFST were considered as “concept tags” indicating the users’ intentions, while its output symbols were considered as “action tags” indicating the system’s actions. Concept tags of user inputs were translated into the corresponding system actions by the WFST. While the WFST-based dialogue management (WFSTDM) is basically the equivalent to that of the conventional finite-state automaton, it can be designed with more flexibility since there are many useful operations for WFSTs to combine and optimize [4].

While the WFSTDM itself has expandability and adaptability as mentioned above, not everyone who engages in the development of SDS can understand and make full use of the WFSTDM. Furthermore, not everyone who wants to develop SDS has well-developed ASR/TTS engines and models. WFSTDM Builder provides most of the components and the environment which are required for developing SDS.

2 Outline of WFSTDM Builder

2.1 System design concept

In the development process of spoken dialogue system and other systems which have speech interfaces, many different contributors who have different skills engage from different standpoints. Fig. 1 shows the schematic diagram of WFSTDM Builder. WFSTDM Builder enables such contributors to construct one system individually via network. Developers do not have to make dialogue scenarios, and scenario writers do not have to write action codes for the system actions. Each contributor can just focus on what they should do to construct the system.

Fig. 1 Schematic diagram of network-based spoken dialogue system builder: WFSTDM Builder
WFSTDM Builder

2.2 Each function

WFSTDM Builder provides five main editors and one sample client application; spoken language understanding (SLU) editor, scenario WFST editor, paraphrase editor, database editor, action code editor and MCML client.

SLU

The role of spoken language understanding (SLU) is to convert a user’s spontaneous speech input to a symbol which the system can utilize. In the WFSTDM, SLUs have to be expressed in the WFST format eventually, but users of WFSTDM Builder do not have to describe them with the format. They only have to register the n-word phrases extracted from the transcripts of the related conversational data as a concept tag they want the system to understand, e.g., “Good morning” to greetings. WFSTDM can convert them to a SLU WFST. If slots are designed and slot values are defined as slot tags, they can be extracted as keywords. The SLU WFST is designed as the key-phrase detector that translates sentences including such phrases to the corresponding concept tags. This loose key phrase-matching enables the system to accept phrases which have the same meaning but have a slight difference in how they are expressed spontaneously.

Scenario WFST

With the scenario WFST the system can make reaction to the user input converted by the SLU WFST as the concept tag. Simultaneously, the system state is transferred to a particular state decided by the scenario WFST. In the framework of the WFSTDM, one can make a dialogue scenario that is statistically trained using a sequence of concept tags in the corpus. Although there are alternatives in choosing responses for users, the scenario WFST enables the dialogue system to determine which system action should be taken in response to the user input according to each state of the dialogue.

Paraphrase

If a word can be expressed in more than two different descriptions, they must be dealt with one exclusive word in the system. For example, USA is often called in a different way such as US, the United States and the United States of America, and so on. Once these words are registered as paraphrases for USA, they are unified into one word, i.e., USA, through the all process.

Database

WFSTDM can refer to and deal with the database for the system action. If the system needs the database which consists of many kinds of categories and terms, it is very important that one can edit the database independently against the SLU and the
scenario. In the view point of maintenance of the SDS, the independency of the database is efficient and also effective.

**Action code**

Although *WFSTDM* initially provides some interpreters such as “SPEAK” to output the synthesized speech, “GO_URL” to display the web page, “DB” to refer to the parameter of the relative database and other sample actions, the developers can register their own action codes to make new system actions. The action codes should be written in C/C++.

**MCML client application**

*WFSTDM Builder* provides a sample client application for iOS ©Apple Inc. (Fig. 2). The client can connect to the servers with MCML protocol. MCML is a communication protocol standardized in ITU-T for network-based speech to speech translation under the activity of U-STAR (see the U-STAR web page http://www.ustar-consortium.com/). *WFSTDM Builder* adopts MCML as its communication protocol, and communicate with all servers related to this protocol.

![Fig. 2 A snap shot of MCML Client for iPhone](image)

### 3 How to use *WFSTDM Builder*

Fig. 3 shows the interface of the *WFSTDM Builder*. All users can connect each server via this interface on web browsers. The procedure is very simple.
WFSTDM Builder

Fig. 3 Initial screen of WFSTDM Builder. All steps for constructing SDS are instructed step by step.

Fig. 4 SLU editor screen of WFSTDM Builder.

Fig. 5 Scenario editor screen of WFSTDM Builder.

STEP 1: Select language and Open project

A login account of WFSTDM Builder is issued for all users. After login, the first step is to select main target language. All steps the users have to do are instructed by the arrows (see Fig. 3). The next step is to open or create a project for constructing a dialogue management including components such as SLU and scenario WFST. In case of creating a new project, the user can reuse the sample project such as a guidance system for tourists and a tutor system for cooking.
STEP 2: Edit SLU

In constructing SLU components, users do not have to make SLU in a WFST format. First, the concept tag or the slot tag should be registered. These tags are handled as symbols in the dialogue scenario and system action. Then, the user only has to register the possible expressions for one concept or slot (fig. 4).

STEP 3: Edit scenario WFST

The dialogue scenario should be described in a WFST format. Based on the scenario, the user just have to register transitions which consist of the source-state number, destination-state number, user concept as input, and system action as output. The user can optionally add weight to each transition for the probabilistic transitions. We are improving the builder so as to display the transition network.

STEP 4: Run WFSTDM server

By simply pressing the “Build” button, dialogue management WFST will be composed from the SLU WFST and scenario WFST, and the WFSTDM server will be started successively.

STEP 5: Test the scenario with Java client

The users can instantly test the dialogue scenario they have built, within the builder either with audio and text. Furthermore, the users can also test the scenario with a MCML client on the iPhone (Fig. 2).

4 Conclusion

In this paper, we introduced the network-based rapid SDS constructing tool WFSTDMBuilder and its usage. We are now preparing the SDK of WFSTDMBuilder so that everyone can use this tool by around March 2014.

References