A Spoken Dialogue System Based on FST and DBN

Lichun Fan[1,⋆,⋆⋆], Dong Yu[2], Xingyuan Peng[1], Shixiang Lu[1], and Bo Xu[1]

[1] Interactive Digital Media Technology Research Center
Institute of Automation Chinese Academy of Sciences, Beijing, China
[2] International R&D Center for Chinese Education
College of Information Science, Beijing Language and Culture University, China

Abstract. Natural language understanding module and dialogue management module are important parts of the spoken dialogue system. They directly affect the performance of the whole system. This paper proposes a novel method named action-group finite state transducer (FST) model to cope with the problem of natural language understanding. This model can map user utterances to actions, and extract user's information according to the matched string. For dialogue management module, we propose dynamic Bayesian network (DBN) model. It can reduce the demands for the corpus compared with Markov decision process (MDP) model. The experiments on the action-group FST model and DBN model show that they significantly outperform the state-of-the-art approaches. A set of subjective tests on the whole system demonstrate that our approaches can satisfy most of the users.

Keywords: spoken dialogue system, natural language understanding, dialogue management, FST, DBN.

1 Introduction

Spoken dialogue systems abstract a great deal of concern from its appearance in the 1990s. In the past two decades, spoken dialogue systems developed rapidly. However, due to the difficulties of natural language understanding and dialogue management, most spoken dialogue systems still stay in the laboratory stage.

As the prior part of the dialogue management, natural language understanding plays an important role in spoken dialogue system. In previous studies [1,2], the user utterance is usually mapped to the system state vector, and the dialogue strategy depends only on the current system state. Natural language parsing is relatively simple keyword matching or form filling based on syntax network. These natural language understanding methods have a poor ability in describing language, and these mapping methods will lose a wealth of language information in user utterances.

⋆ Contact author: `lichun.fan@ia.ac.cn`

⋆⋆ This work was supported by the National Natural Science Foundation of China (No. 90820303) and 863 program in China (No. 2011AA01A207).

As the center of spoken dialogue system, dialogue management has been a hot academic research area for many years. Early spoken dialogue systems are entirely designed by experts. The systems usually employ a finite state machine model [3] including a large number of artificial designed rules. These artificial rules are written in a very good expression and they describe the specific applications correctly, but the preparation and improvement of the rules will become increasingly difficult when the dialogue task becomes complex. In recent years, data-driven based dialogue management model gradually attracts the researchers' attention. This kind of dialogue management model obtain knowledge through the annotated dialogue corpus automatically. Dialogue management mathematical model based on Markov decision process (MDP) is proposed in [4], and the reinforcement learning (RL) is proposed to learn the model parameters in the same study. Studies in [5] propose a partially observable MDP (POMDP) dialogue management model. These models have become hot topics of the dialogue management approaches since they are proposed.

However, there remain many problems when these models are used in the actual applications. Firstly, the system uses a reinforcement learning mechanism. Interaction with the actual users is very time-consuming, and is hard to achieve the goal. Many systems get the optimal parameters from the interaction with the virtual users, so there are some difference between the optimal strategy and the real dialogue. Secondly, the dialogue models obtained by this method lack of priori knowledge of dialogue, so they are difficult to be controlled.

In this paper we propose the finite state transducer (FST) model to solve the natural language understanding problem. The FST model maps the user utterances to user actions, and extracts the attribute-value information in user utterances. For dialogue management problem, we propose dynamic Bayesian network (DBN) model. It generates the dialogue strategy not only considering the system state, but also depending on the user actions.

The paper is organized as follows: the action-group FST model and the details of using action-group FST model to detect the user actions are presented in section 2. Section 3 describes the dynamic bayesian network and dialogue management based on DBN model. Section 4 presents our spoken dialogue system and the performance of the two modules. Finally, conclusions are presented in section 5.

2 User Action Detection Based on Action-Group FST

2.1 User Action-Group FST Model

Each utterance corresponds to a single action. We annotate the utterances by action labels, and mark out the information contained in the utterances. Then we collect the utterances which have the same action label together to generate an action-group. We use the "determinize" and "minimize" operation described in the openFST tools [6] to unite and optimize the action-group FST. Fig.1 shows the building process of an action-group FST for a particular action-group.

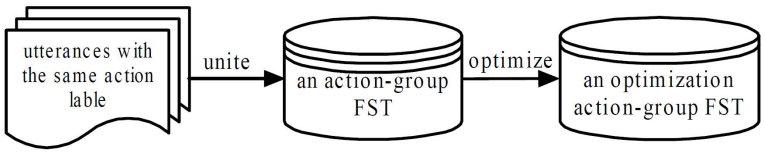

utterances with the same action lable
unite
an action-group FST
optimize
an optimization action-group FST

Fig. 1. The building process of action-group FST

It is unrealistic to establish a multiple action-group FST by many action-groups which are built through dividing all of the daily utterances into different action groups. In this paper, the proposed action-group FST is only used in the fields identified by the dialogue scenes. The utterances used for building the action-group FST are collected in these particular fields.

These utterances are labeled in two stages, the first stage only marks which action-group the utterances belong to, while the second stage needs to mark out the information contained in every utterance. We use attribute-value pairs [7] to describe the information contained in utterances, such as “currency=dollar”. The purpose of the second-level label is to parse the user utterances automatically according to the string sequence in the FST model that matches the user utterance properly. If the action-group is labeled according to domain knowledge, the model will be too sparse, and the portability will be poor. However, it is unable to accomplish the dialogue management tasks if the action-group is labeled based on sentence structure. Therefore, we design the action-group according to [7]. Table 1 lists the major action-groups that we used in our system.

Table 1. Action-group list

Action	Description
hello(a=x,b=y)	open a dialog and give information a=x, b=y,. . .
inform(a=x,b=y)	give informationa=x, b=y,. . .
request(a,b=x,. . .)	request value for a given b=x . . .
reqalts(a=x,. . .)	request alternative with a=x,. . .
confirm(a=y,b=y,. . .)	explicitly confirm a=x,b=y,. . .
confreq(a=x,. . . ,d)	implicitly confirm a=x,.. and request d
select(a=x,b=y,. . .)	select either a=x or a=y
affirm(a=x,b=y,. . .)	affirm and give a=x, b=y,. . .
negate(a=x)	negate and give further info a=x
deny(a=x)	deny that a=x
bye()	close a dialogue

2.2 User Action Detection Based on Action-Group FST

We hope to find the most similar word sequence as the user utterance in the action-group FSTs. Then we consider the user utterance belongs to the action-group where the word sequence is in. In this paper, the edit distance is used to measure the degree of similarity between the two word sequences. The decoder's work is to find the optimal word sequences' path in action-group FST that can make the edit distant minimum.

The most common method used to calculate the edit distance is the Wagner-Fischer algorithm [8]. This is a string to string edit distance calculation method, and its core idea is dynamic programming. We improve the Wagner-Fischer algorithm to compute the edit distance between the string and the network. The algorithm is also based on the idea of dynamic programming. It calculates the edit distance between user word string and each node in the FST respectively. The pseudo-code for this dynamic programming process is shown in Table 2.

Table 2. Improved Wagner-Fischer algorithm

```
inputs:  W  is a word string
         F  is a FST model with M states
Initial:Compute $minEDcost(n_0, W)$
For each state $n_i$ in $F$, $0 < i \leq$
    Compute $cost(n_{i-1,n_i})$
    $minEDcost(n_i, W) = min(minEDcost(n_{i-1}, W) + cost(n_{i-1}, n_i))$
Output:  S  is the road that has the least $EDcost$ with $W$ in $F$
```

In table 2, $minEDcost(n_i, W)$ represents the accumulated minimum edit distance from state 0 to state i and $cost(n_{i-1}, n_i)$ represents the increased edit distance from state $i-1$ to state i.

When the user utterance matches a string sequence which has the smallest edit distance with the utterance from an action-group FST, we assume the user utterance belongs to the action-group. At the same time we can automatically extract the attribute-value information in user utterance according to the matched string.

3 Dialogue Management Based on DBN

3.1 DBN

Dynamic Bayesian network (DBN) is a probabilistic graphical model that can deal with the data with timing characteristics. It is formed by adding time information in the conventional Bayesian network. In the DBN, the topology of the network in each time slice is the same. Time slices are connected through a number of arcs which represent the relationship between the random variables in different time slices. Fig.2 shows the topology of a DBN model.

Bayesian network learning consists of two categories: parameter learning and structure learning. Parameter learning is to estimate the conditional probability distribution of each node by the training data when the network structure is known. The structure learning, also known as model selection, is to establish the network topology through training data. Structure learning problem is very

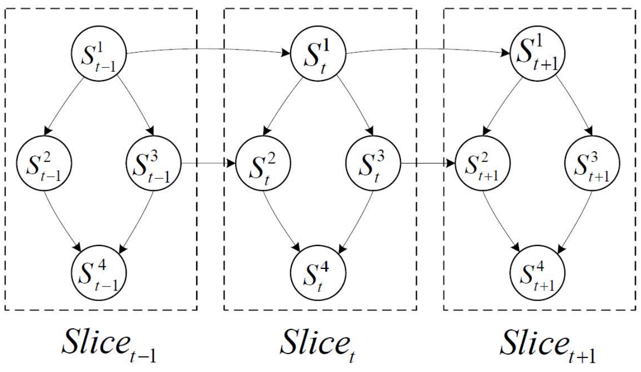


Fig. 2. The topology of DBN

complex, so we manually define the network structure and mainly consider the problem of parameter learning in this paper. Under the conditions of a given network structure, parameter learning can be divided into two categories according to the observability of nodes.

If all nodes can be observed, the model parameters can be obtained directly by a large number of data samples. However, in most cases the data we collected is not enough, then the Dirichlet distribution can be used as the prior distribution of parameters. We update the model through the "evidence" until we get the ultimate posterior probability parameters. We use the notation B^h to represent the Bayesian network topology, and $X = x_1, \ldots, x_n$ to represent a set of n discrete random variables in the network, where each x_i has r_i possible values $x_i^1, x_i^2, \ldots, x_i^{r_i}$. Then the joint probability distribution of X can be shown as the following equation:

$$P(X|\theta_s, B^h) = \prod_{i=1}^{n} P(x_i|parents(x_i), \theta_i, B^h) \tag{1}$$

where $p(x_i|parents(x_i), \theta_i, B^h)$ is the local probability distribution that corresponding with node x_i, and $\theta_s = (\theta_1, \ldots, \theta_n)$ is the parameters of the network model. We assume the training data set $D = \{d_1, \ldots, d_N\}$ has a total of N samples, and each sample includes all the observable nodes. Then the Bayesian network parameter learning problem can be summarized as follows: calculate the posterior probability distribution $P(\theta_s|D, B^h)$ when given the data set D. We get the following equation:

$$\theta_{ijk} = P(x_i^k|parents^j(x_i^k), \theta_i, B^h) \tag{2}$$

where $\theta_i = ((\theta_{ijk})_{k=2}^{r_i})_{j=1}^{qi}$ and $q_i = \prod_{parents(x_i)} r_i$. We define $\theta_{ij} = (\theta_{ij1}, \ldots, \theta_{ijr_i})$ and assume that parameter vectors θ_{ij} are independent with each other, then the posterior probability of the model parameters is shown as the following equation:

$$P(\theta_s|D, B^h) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} p(\theta_{ij}|D, B^h) \tag{3}$$

The prior distribution of Parameters θ_i is Dirichlet distribution, and then we get the follow equation:

$$P(\theta_{ij}|D, B^h) = Dir(\theta_{ij}|\alpha_{ij1} + N_{ij1}, \alpha_{ij2} + N_{ij2}, \ldots, \alpha_{ijr_i} + N_{ijr_i}) \tag{4}$$

where N_{ijk} is the number of occurrences of $x_i = x_i^k$,$parents(x_i) = parents^j(x_i)$ in the data set D, and α_{ijk} is the number of occurrences of the events in priori, and $\alpha_{ij} = \sum \alpha_{ijk}$, $N_{ij} = \sum_k N_{ijk}$. Finally, according to equation 3 and equation 4, we obtain the following equation[9]:

$$P(X|D, B^h) = \prod_{i=1}^{n} \frac{\alpha_{ijk} + N_{ijk}}{\alpha_{ij} + N_{ij}} \tag{5}$$

When there are hidden nodes in the network, the parameters of the model can be estimated via the EM algorithm. In the E-step the expectations of all nodes are calculated, while in the M-step these expectations can be seen as observed, and the system learns new parameters that maximize the likelihood based on this. After a number of iterations, the system will converge to a local minimum.

3.2 Dialogue Management Based on DBN

In this paper, we do not use the complex state space in the spoken dialogue system. We just put the user action into the dialogue management model to affect the dialogue action [10]. Similarly, we assume that the dialogue process has the Markov property, and then get the following equation:

$$P(a_t|S_1^t, A_1^{t-1}, U_1^t) = P(a_t|s_t, u_t) \tag{6}$$

where S_1^t, A_1^{t-1}, and U_1^t represent the system state, the system action, and the user action that from the initial moment to time t, respectively. s_t, a_t, and u_t represent the system state, the system action, and the user action at time t. The DBN model will take the system input at every time into account, so we add the user action into every time slice in order to make the model depend on the user action. The system action at time t depends on the system state s_t and the current user action u_t. Meanwhile, the current user action u_t as a system input also directly affects the system state s_t. We retain the transfer connection arc between the states in adjacent time slices. The DBN model structure described above is shown in Fig.3.

According to the above description, the DBN model consists of four parts: the system state space S, system dialogue action space A, the DBN network topology B and the user action space U. We collect a series of dialogue corpus as "evidence" to learn the model parameters and infer the optimal system action.

If the system state vector s_t is observable, we can update s_t according to the attribute-value information that extracted by the action-group FST model.

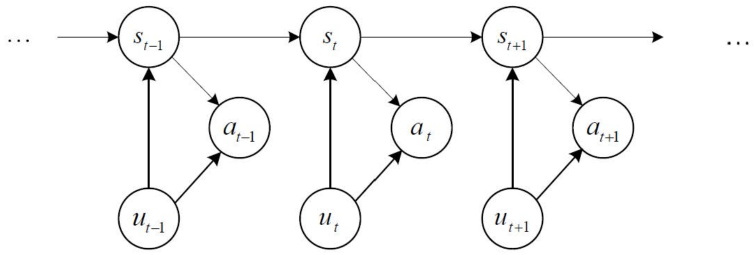


Fig. 3. The DBN Model with user action

Then we can estimate the model parameters directly from the data according to the parameter learning method described in section 3.1. To inference the system action, we directly calculate the joint conditional probability distribution of the system action under the conditions of the system state vector s and user action vector u, and take the maximum probability of action as the current system action. This idea is described by the following equation:

$$\hat{a}_t = \arg\max_{a_i \in A} P(a_i | s_t, u_t) \tag{7}$$

If the system state vector is unobservable, we have to use the EM algorithm to estimate model parameters from the "evidence". The current system action is calculated by the following equation:

$$\begin{aligned} \hat{a}_t &= \arg\max_{a_i \in A} P(a_i | u_t) \\ &= \arg\max_{a_i \in A} \sum_k P(a_i | u_t, S = s_k) P(S = s_k | u_t) \end{aligned} \tag{8}$$

4 System Structure and Experimental Results

The previous content describes the modeling method of natural language understanding module and the dialogue management module. This section presents the overall design of the spoken dialogue system, and provides several experiments to verify the performance of each module. Finally, the experimental results are analyzed.

4.1 Overall Structure of the Spoken Dialogue Systems

A typical spoken dialogue system not only includes the natural language understanding (NLU) module and the dialogue management (DM) module, but also contains the automatic speech recognition (ASR) module, the text to speech (TTS) module, and the natural language generation (NLG) module. For the spoken dialogue system in particular scene, it also contains a database or network database for inquiry. Our spoken dialogue system structure is shown in Fig.4.

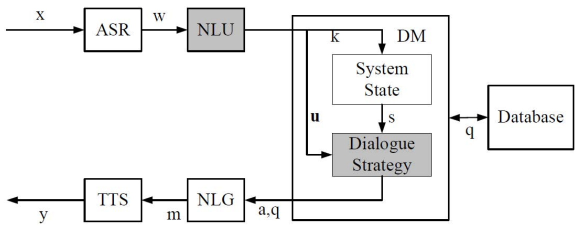


Fig. 4. The structure of our spoken dialogue system

The implementation of the ASR module and the TTS module in Fig.4 uses the open source HTK tools [11] while the NLG module uses template filling [3] method. We use mysql to bulid the database and use query to build communication between the database and DM module. The NLU module and dialogue strategy in the shaded blocks are the focus of our studies. This system receives the user voice x, and uses the ASR module to convert it to the text sequence w. Then the action-group FST model in the NLU module maps the word sequence w to the user action u and attribute-value information k. The dialogue management module has two functions: to maintain the system state and to give the dialogue strategy. We use form filling method to represent the system state [1]. The dialogue management module uses the attribute-value information k to update the system state s, and then uses the DBN model to calculate the dialogue strategy based on the system state s and the user action u. If a user requests for information, the dialogue management module needs to interact with the database to query information q. Then the dialogue management module sends the system action a and the information q to the NLG module where the information will be integrated into the response text m. Finally, the synthetical voice y generated by the TTS module is passed to the user.

4.2 Experimental Results

Experiments on Action-Group FST Model. In this section, we will present the action-group FST model built by openFST toolkit. The annotated data is divided into the training and testing set. The training set consists of 100 dialogues, with a total of 671 rounds data, while the attribute-value pairs in user words are 846. The testing set consists of 20 dialogues, with a total of 138 rounds data, while the attribute-value pairs in the user words are 185. In the annotating process, words that cannot be listed have an alternative, such as the amount of money in the currency exchange scene is replaced by “amount”. We use the classification accuracy of user action and the precision, recall and F-measure of the attribute-value pairs in user utterance to evaluate the action-group FST model.

The experiments are divided into two groups. We test the effect of the action-group FST model and the traditional keyword matching respectively. The rules of keyword matching method are studied from the training set only, and they are used to identify user action and extract user information. The results are shown in Table 3.

Table 3. The performance of action-group FST Model

	classification	attribute-value pairs extraction		
	precision	**precision**	**recall**	**F-measure**
action-group FST model	92.75%	92.05%	87.57%	89.75%
keyword matching	90.58%	96.25%	83.24%	89.27%

The experimental results prove that the action-group FST model outperforms the keyword matching method. The keyword matching method has better precision, but its recall is worse than that of the action-group FST model. The overall effect of the action-group FST model has a slight improvement compared with the manual keyword matching method. However, the keyword matching method needs experts to spend much time to write the rules, and this work will become very difficult when the amount of data increases. The action-group FST model will perform better if the amount of data increases, and it has good portability. In summary, the action-group FST model has more advantages.

Experiments on DBN Model. This section presents the DBN model built by the BNT toolkit [12]. The annotated data is also divided into the training and testing set, and the data size is the same as described above.

In the training phase, 671 rounds of dialogue data are sent to the DBN model by sequence to estimate the model parameters. The priori parameters of the initial model are obtained by the Dirichlet distribution. In the testing phase, we send the user utterances to the DBN model by their sequence in the dialog. The difference between the system output action and real data is recoded. When the system generates a wrong action, we send the user utterance by sequence just as there is not any error. Finally, we use the precision of the system output to evaluate the performance of the DBN model.

In this section, we test the DBN model and the MDP model separately, and the experiments on the DBN model are divided into three groups. We assume all nodes in the DBN model are observable in the first experiment. Therefore, there are three variables in the training data. They are the user action, the system state and the system action. The testing data include only the user action and the system state, and the DBN model infers the most suitable system output action according to these two variables. In the second experiment, we set the system state in the previous time slice as evidence both for training and testing. Therefore, the DBN model infers the most suitable system output action based on three variables. We hide the system state in the third experiment. Therefore, there are only the user action and the system action in the training data. We use EM algorithm to estimate the parameters of the DBN model and the DBN model infers the most suitable system output action based only on the user action.

Table 4. The performance of DBN model

	DBN			**MDP**
	$P(a_t \mid s_t, u_t)$	$P(a_t \mid s_{t-1}, s_t, u_t)$	$P(a_t \mid u_t)$	$P(a_t \mid s_t, u_t)$
precision	90.58%	89.13%	88.41%	81.88%

Table 4 shows the results of these experiments. The last experimental result in Table 4 is obtained by the MDP model [4] on the same data set described above. In the training phase, the award value of each step is set to -1. If the dialogue ends with satisfaction, then the system rewards 20 for the whole dialogue. There are about 10 rounds in each dialogue, so the whole dialogue gets a positive award sum if the dialogue ends with satisfaction, otherwise it gets a negative award sum. The final jump probability is obtained according to the normalized award value.

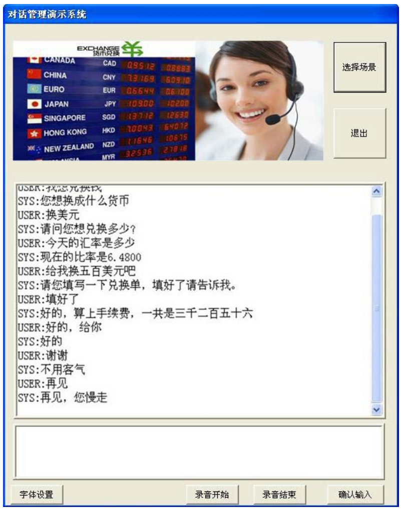
From the results shown in Table 4, we can see that the DBN model in the first experiment outperforms the model in the second experiment. This proves that the simple DBN model performs better than the complex model in short dialogue. In the third experiment, the performance degrades significantly when the system state is hidden. This means the system state is very important for DBN model. In related work [7], more complex method is used to describe and maintain the system state, which is consistent with our experimental results. In addition, the performance of the MDP model is poor. This is probably because the MDP model is to get the global award maximum while the DBN model is to seek a single round of local optimal. In the case of limited training data, it is difficult to get the optimal solution for the MDP model. This conclusion is consistent with the related work in [4].

4.3 The Overall Performance of Our Spoken Dialogue System

Natural language understanding and dialogue management, as two of the most important modules in spoken dialogue systems, have good performance separately through the action-group FST model and the DBN model. However, a conclusion concerning the performance of the whole system cannot be drawn easily through these results. A subjective experiment on our overall system proves that the system can basically meet user's needs. A survey of the user of our system is shown in Table 5. The results show that the performance of the system is related to the users, and researchers' test results are much better than those of the other users. The main reasons for the errors are that the user utterances is beyond the training data, and ordinary users are more likely to say words outside the training set. Fig.5 is an instance of the spoken dialogue system interaction with human in a currency exchange scene.

5 Conclusion

This paper focuses on the natural language understanding and dialogue management in spoken dialogue system. To cope with these tow problems, we propose



USER:我想兑换钱
SYS:您想换成什么货币
USER:换美元
SYS:请问您想兑换多少?
USER:今天的汇率是多少
SYS:现在的比率是6.4800
USER:给我换五百美元吧
SYS:请您填写一下兑换单，填好了请告诉我。
USER:填好了
SYS:好的，算上手续费，一共是三千二百五十六
USER:好的，给你
SYS:好的
USER:谢谢
SYS:不用客气
USER:再见
SYS:再见，您慢走

Fig. 5. An instance of our spoken dialogue system

Table 5. A user survey of the spoken dialogue system

user types	**total dialogues**	**dialogues end with satisfaction**
researcher	30	27
ordinary user	30	22

the action-group FST model and the DBN model. The advantages of these two models include: (1) the model can work on a small-scale data set, while the model will perform better on a large-scale data set; (2) these two models have good portability because there is little artificial work in model designing.

Although the action-group FST model and DBN model show good performance in some specific fields of spoken dialogue systems, there remains much to improve. If the concept of word similarity is added into the action-group FST model, the edit distance calculation results of the synonyms will be a decimal. Then the action-group FST model can perform better when matching user utterances. In addition, the form filling method used for representing the system state cannot meet the system's needs when the system becomes complex. Furthermore, the DBN structure design and parameter learning will be more complex when the system state becomes a high-dimensional vector. These issues will be addressed in future work.

References

1. Seneff, S., Polifroni, J.: Dialogue Management in the Mercury Flight Reservation System. In: Proceeding ANLP/NAACL Workshop on Conversational Systems, pp. 11–16 (2000)

2. Pietquin, O.: A probabilistic framework for dialog simulation and optimal strategy learning. IEEE Transactions on Audio, Speech, and Language Processing, 589–599 (2004)
3. Pieraccini, R., Levin, E., Eckert, W.: AMICA: the AT&T mixed initiative conversational architecture. In: Proceedings of the European Conference on Speech, Communication and Technology, pp. 1875–1878 (1997)
4. Levin, E., Pieraccinin, R., Eckert, W.: A stochastic model of computer-human interaction for learning dialog strategies. IEEE Transactions on Speech and Audio Processing, 11–23 (2000)
5. Williams, J.D., Young, S.: Partially observable Markov decision processes for spoken dialog systems. Computer Speech and Language, 393–422 (2007)
6. Allauzen, C., Riley, M., Schalkwyk, J., et al.: OpenFst: A General And Efficient Weighted Finite-State Transducer Library. In: Proceedings of International Conference on Implementation and Application of Automate, pp. 11–23 (2007)
7. Young, S.: Still Talking to Machines (Cognitively Speaking). In: Interspeech, pp. 1–10 (2010)
8. Wagner, R.A., Fischer, M.J.: The String-to-String Correction Problem. Journal of the ACM, 168–173 (1974)
9. Heckerman, D.: A tutorial on learning with Bayesian networks. SCI (2008)
10. Yu, D.: Research on Error Detection and Human-Computer Dialogue for Error Correction in Cross-language Conversation, pp. 50–56. Institute of Automation, Chinese Academy of Sciences, Beijing (2011)
11. Young, S., Evermann, G., Gales, M., et al.: The HTK Book (for HTK Version 3.4). Cambridge University Engineering Department (2009)
12. Murphy, K.: The Bayes Net Toolbox for Matlab. In: Computing Science and Statistics: Proceedings of the Interlacel (2001)