Spoken Dialog System

Emanuele Viglianisi (187270)
emanuele.viglianisi@studenti.unitn.it

Abstract

In this report we show how to design a spoken dialog system using Web Speech API and the FST models that we have created for the first midterm project. The report will describe how the system works, focusing on some important aspects of a dialog system such as the implementation of error recovery, confirmation recovery, the use of a multi-modal system etc.

1 Introduction

Spoken dialog system are software that use speech recognition and speech synthesis to interact with the user, and understand and elaborate a user's request in order to provide a service. The aim is making the interaction with the user as natural as possible, by simulating the behaviour of a human conversation. We have developed a spoken dialog system in the movie domain which the user can ask for information about a movie such as the director, the cast etc. The system will interpret the question and answer to the user with the requested information. We use the Web Speech API to handle user interaction, and FST models to interpret the question. The requested information are then retrieved from a database. To make the interaction more realistic and, to correct errors of the system in interpreting the request of the user, a set of dialog rules has been applied: such as turn taking, grounding, confirmation and error recovery.

2 Dataset

The dataset used in all the experiments is Microsoft NL-SPARQL Dataset (Hakkani-Tur et al., 2014). NL-SPARQL is a dataset of natural language (NL) utterances to a conversational system in the movies domain. The dataset contains utterances like "when was men in black released" which are like the ones that the users ask to our system. The dataset is divided into:

- training set of **3338** utterances
- test set of **1084** utterances.

The training set contains, for each utterances, the words associated with the corresponding lemma, part-of-speech tag and the concepts expressed in IOB notation (B-movie.name, I-movie.name etc.).

3 Description of the system

The dialog system is composed of a frontend and a backend application.

3.1 Frontend application

The frontend application was developed in HTML/Javascript and handles the interaction with the user. It uses the javascript API WebSpeechAPI (Web) that includes the components for Speech recognition and Speech synthesis. Using these two components the application listen for question from the user, gets the most probable transcription and the corresponding confidence and sends them to the server to be analysed using an HTTP request. Then, the client waits for a response of the server.

3.1.1 server interaction

Each response contains the current status of the dialogue and a content of the reply. The client can receive three types of replies: ask_target, ask_intent, final. On the basis of the current status and on the server's reply, the client behaves in different ways. For example, if the server requires a confirmation, then the client asks the user and sends back the user's input to the server. At the end, the server sends a reply of type "final" that contains the answer or, in case there is no a valid answer for that question, it contains an error.

3.1.2 user interaction

When a response from the server is received, the client can continue the interaction with the user in two ways:

- ask_intent: asks the user to specify or repeat what he is interested in. The server provides a list of possible *intents* which are listed to the user (es. did you ask for director or actor or something else?)
- ask_target: asks the user about the target. First the client asks whether the target is correct or not, and then, ask for a confirmation of the predicted tag. (es. is star wars a movie?).

If the system can't handle the question, it will redirect the user to a Google search.

The system respects some of the good practice of the spoken dialog system:

- *dialog structure*: The dialog has a fixed structure. First of all, the system opens the conversation with a personal identification "Hi, what is your question" and ends it with "The answer to your question is". In the middle, if the server has some requirements, the system may ask for confirmation or to specify, or repeat the intent or the target of the question.
- *turn taking*: The system receives the task from the server and asks the user about it in speech and then listens for a while waiting for the answer of the user. Since in natural dialog the user may want to interrupt who is talking, the system stays in listening also while it is speaking to the user. However, since the Speech Recognition is not smart enough to discriminate between the voice of the users and the voice of the synthesizer that comes out from the PC's speakers, this feature is optional and recommended only with the use of headsets.
- *grounding*: the user have to know whether his requests succeed or not. The system uses *acknowledgement* and *demonstration* so that the user can figure out what the system understands and what it doesn't. In the system's sentences are frequently used "Ok", "I was wrong", "Got it" to show *acknowledgement* and "Ok, I'm looking for the director" in order to demonstrate what the system has understood.
- *multimodal system*: since the user has to use the application inside a internet browser, he must be able to interact using both speech and web page controls. We have also developed a HTML interface to show the user the status of the dialog: user input, system outputs, possibilities, suggestion for the response etc. For instance, if the system asks the user to choose from a list of possibilities, it is shown as clickable HTML list.

3.2 Backend application

The backend application was developed in PHP. When it receives a *start request* from the client, containing the question of the user, it executes a fixed pipeline:

1. Compile the input string into a fst
2. Extract Concepts from the input
3. Extract user requested information (intent) using a FST-based Naive Bayes classier
4. Check confidences of the extracted information and, if necessary, ask confirmation to the user
5. Build the SQL query and retrieve information from an SQL database

3.2.1 Concept tagging

The user input has to be analysed in order to extract the information required to retrieve the answer from the database. The training set was used to create an uni-gram concept tagger and a language model that, composed together with the user input, will generate all the possible concept tagging for the utterance. The result is is passed through the command *fstprintstrings* to output all the possible strings and, at the end, the first n-best taggings are taken for the analysis.

$$inputString_{fsa} \circ conceptTagger_{fst} \circ LM_{fst}$$

For more information about the dataset and the FST models, refer to the mid-term project about Spoken Language Understanding (SLU) module (Mid).

3.3 fstprintstrings limitations

The fact the command fstprintstrings produces all the possible strings from the resulting final state machine of the composition represents a problem

when we have to deal with large utterance. Producing all the possible strings will take up to 6GB of RAM for a 12 words utterance and this represent a limitation for our system. A possible solution could be the use of the command *fstshortestpath* to consider only the shortest path instead of all the possible path. The result of this method diverges from the original method, but the best **n-results** are still valid and acceptable.

3.3.1 Choice of the thresholds

Tagging and prediction operations also output the confidences. Confidences are measures that states how probable is the result to be correct. Following on from the best model of the SLU project, where the cost associated with each transition is the negative log of the probability $P(word, lemma, pos|concept)$, the confidence for the concept tagging is computed as follows:

$$\frac{probability\_path}{sum\_of\_probabilities} * ASR\_conf$$

Where ASR_conf is the confidence from the Speech Recognition API. If the confidence is low, the result will be far to be correct. Instead of returning an incorrect answer to the user, the idea is to decide thresholds below which it is better to engage a conversation with the user asking for confirmation of the extracted tags and/or the predicted intent. Considering only the results with the highest confidence as *best_tagging*, the threshold for acceptance is computed using the utterances in the test set with the following formulas:

$$thrs.\_slu = mean(best\_slu\_tagging) = 0.89$$
$$thrs.\_intent = mean(best\_intents) = 0.93$$

If the confidence is below the respective threshold means that the system is not sure enough, and requires confirmation from the user.

The system doesn't have a rejected threshold because, during the confirmation request, the user is always able to correct the tagging or the intent.

4 Evaluation and Conclusion

To evaluate the system we have manually annotated the results obtained using a small test set of utterances.

Table 1: Intents - Baseline

correct/predicted	movie	director	actor	r.date
movie	**2**	0	0	0
director	1	**3**	1	0
actor	2	0	**1**	0
release date	1	0	0	**3**

Table 2: Intents - Error recovery

correct/predicted	movie	director	actor	r.date
movie	**2**	0	0	0
director	0	**5**	0	0
actor	0	0	**3**	0
release date	1	0	0	**3**

The tables 1 and 2 represent the confusion matrices of intents and show how much the prediction of the intent differs from the effective intent request. The information that we can extract from them are that, although the baseline has already good results, adding error-recovery, the system has a little improvement in performances, and it does not mis-predict the intent *movies* with respect to all the other intents. When error-recovery is employed all predictions with confidence lower than the threshold were corrected. There were however some mispredictions with confidence over the acceptance threshold. The same is also true for the SLU tagging where the baseline correctly recognises the target 8 times out of 10.

References

Midterm project: Spoken language understanding module. https://github.com/emavgl/slu_movie_domain.

Webspeechapi. https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html.

Dilek Hakkani-Tur, Asli Celikyilmaz, Larry Heck, Gokhan Tur, and Geoff Zweig. 2014. Probabilistic enrichment of knowledge graph entities for relation detection in conversational understanding. In *Proceedings of Interspeech*. ISCA - International Speech Communication Association, page 1. research.microsoft.com/apps/pubs/default.aspx?id=219835.

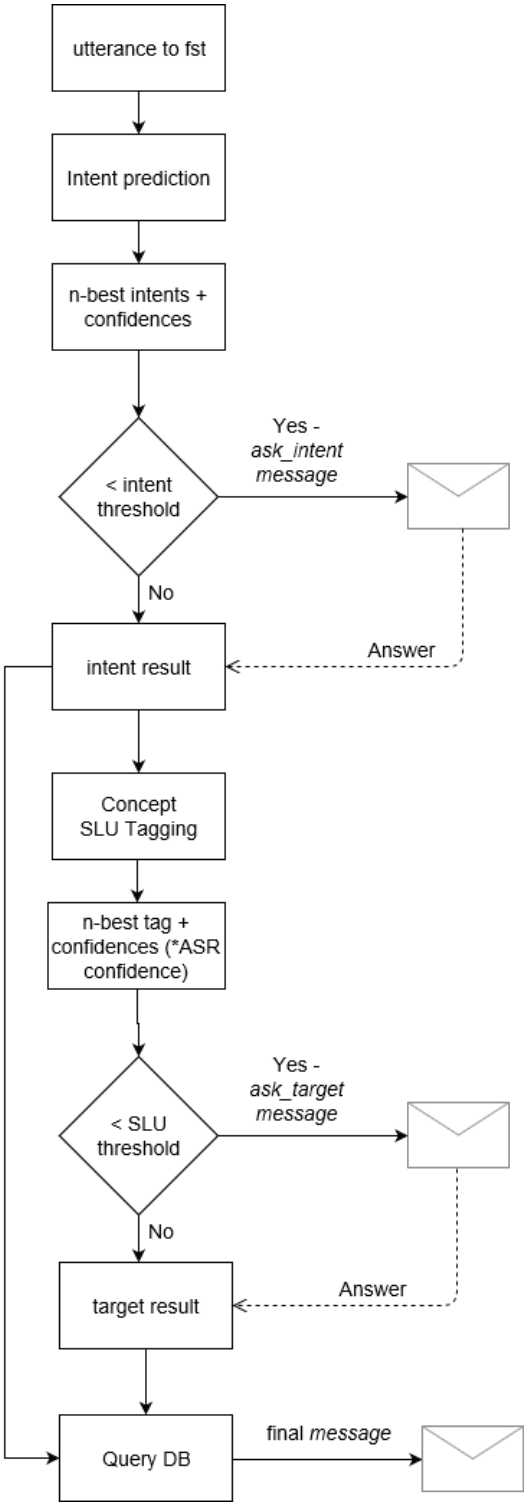

Figure 1: Server Pipeline Schema

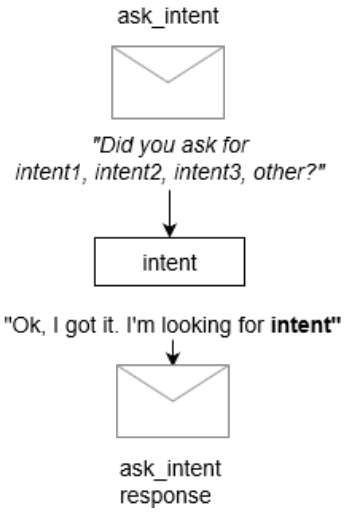

Figure 2: client: ask_intent schema

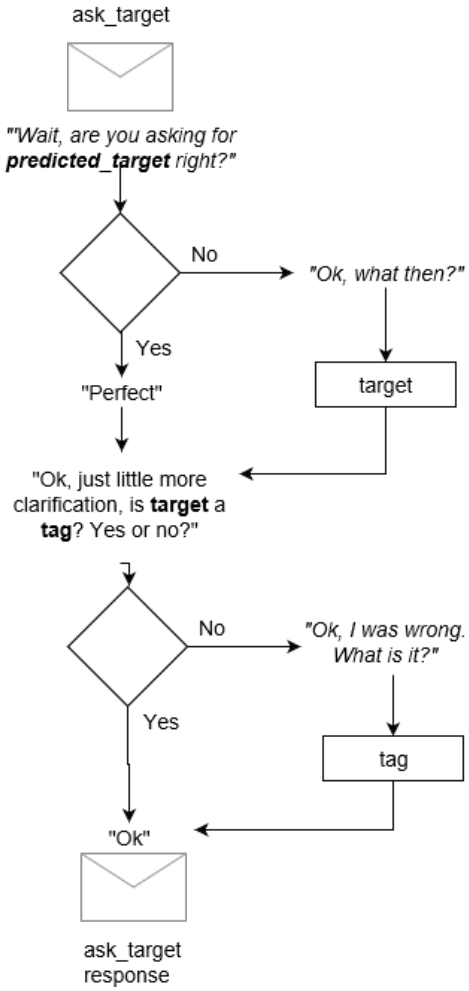

Figure 3: client: ask_target schema