Language Understanding Systems
Spoken Dialogue System Project

Nicola Garau (189086)
nicola.garau@studenti.unitn.it

Abstract

The goal of this project was to provide a Spoken Dialogue System (SDS) in the Movie Domain, capable of interacting with the user by vocal input and output. The SDS described in this document is based on my first part of the course assignment, FST & GRM Tools for SLU (Concept Tagger trained with the Kneser-Ney smoothing algorithm, 9-gram).
Additional requirements for the project were to implement and evaluate Error Recovery by using confidence scores, thresholds and prompts.

1 Description of the system

In this section will be briefly explained how the system works, taking a look at the dataset used, along with the tool itself and some problems faced during the development.

1.1 Dataset

The dataset used is the same Movie Domain Microsoft NL-SPARQL dataset, split in train and test utterances, tagged with Part Of Speec (POS) tags and Concept Tags using IOB notation. To get more information about the dataset's analysis and explore some problems linked to it, please take a look at the first report that you can find linked here or on my Github page.

1.2 Environment

The development environment was configured as follows:

- Ubuntu 16.04.2 LTS
- Google Chrome 58.0.3029.96 (64-bit)
- PHP 5.6.30 (cli)
- Mysql 15.1 Distrib 10.1.21-MariaDB

1.3 Tool

The tool is composed of a HTML web page that interacts with some PHP files.
The HTML page itself cointains the page structure and style, along with some minor checks on the dialogue flow and all the TTS and ASR logic.

1.3.1 View

From the HTML page (speech.html), the user can send a vocal or textual request to the system. The system answers by using speech and a visual feedback at the same time, thus making the system multimodal (even if very shallow).
In fact, only the first input interation can be textual, while the remaining ones (if needed) must be vocal.

1.3.2 Controller

The main (PHP file) is responsible for all the under the hood logic, including the shunting operations for managing the dialogue session, the interaction with the Movie Database and the checks on the thresholds.
Additionaly, it includes a logging method and is capable of interacting with FstClassifier and FstSlu files to make predictions on the choice of the topic and object of the utterance.

1.4 Model

While for the utterance classification (UC) an existing classifier was provided (FstClassifier.php), for the concept tagging the following parts were taken from the previous project:

- A lexicon file for the Movie Domain
- A finite state transducer for the tagging from word to concept (wordToConcept.fst)
- A language model for the Movie Domain

and fed to the FstSlu.php file for the classification.

1.5 fstprintstrings Problems and possible solutions

During the development and the testing, there were some issues associated with the function fstprintstrings. Besides the compilation issues, the main problem with fstprintstrings is that its code is written is such a way that the more are the words in an utterance, the more it requires a lot of hardware resources (memory) to process it.
Two main fixes can be applied in this case:

- Setting a memory limit: by default, I decided to include a 2Gb memory limit in the file controller.php by adding a `"ini_set('memory_limit', '2048M');"` line at the top of it. Since the major memory leak happens when trying to call fstprintstrings, the memory limit can be bypassed by adding a `"ini_set('memory_limit', '-1');"` line to the top of the FstSlu.php file.

- A better solution to this issue could be calling fstshortestpath after the composition of the utterance and the wfst and the language model, and after the epsilon removal. This solution can lead to a much faster processing of the utterance and the removal of the memory leak. A contraindication of this solution is that after this process the confidence scores get tweaked a bit, even if it's not clear why.
  But overall the solutions can be considered acceptable. This fix has been included by default in this solution and can be found in the file FstSlu.php (line `"$cmd .= ' | fstrmepsilon | fstshortestpath --nshortest=3 | ';"`).

2 Error recovery

By error recovery we refer to those error handling strategies that come to the rescue whenever an interaction error between the user and the system occurs. In the solution proposed by this document are covered all the errors caused by a wrong utterance pronounciation by the user and by a misinterpretation of it by the system.

2.1 Evaluation

In particular, we can see that misinterpretation errors occur when evaluating the utterance intent or the different fields of the user request (objects). We will see that those kind of errors can be always solved by asking the user for a confirmation each time an error occurs or each time the confidence is too low.

In order to manage those kind of situations with error recovery, there was the need to set at least an acceptance threshold for both the UC confidence and the SLU confidence.

2.2 UC-level Error Recovery

During the initial stages of the testing phase, there were some issues related fo the classification of the utterance topic. They were related mainly to two aspects:

- Wrong user pronounciation
- Wrong classification by the classifier

Some critical scenarios, like:

- Wrong intent predicted but with low confidence value
- Right intent predicted but wit low confidence value

can be easily solved by taking into account the acceptance threshold.
Others, like:

- Wrong intent predicted but with high confidence value
- Right intent predicted but with no match in the database

cannot be solved in a trivial way or cannot be solved at all. The only solution for that kind of scenarios is to improve the classifier algorithm itself or to further populate the database.
Whenever a non-trivial solution was needed, the solution offered by the system presented in this document was to simply search for a possible answer on Google.

2.2.1 UC n-Best choice

Since the predict function of the UC classifier can output a number n of best classified intent predictions but it calculates them all in any case, it was necessary to fix n to speed up the system performances.
The choice of n was made by running the classification for all the utterances in the test file and taking the mean of all the number of predictions above 1% of UC confidence for each utterance. It emerged from this test that a good value for n was 3.

2.2.2 UC Threshold

In a similar wat to the n-best choice, the UC acceptance threshold was fixed by running the classification for all the utterances in the test file and taking the smallest value of UC confidence amongst all the correct predictions.
Once the threshold had been set, the UC-level error recovery strategy adopted was the following:

1. Check if the UC confidence is above the threshold
   - If above, return the intent
2. If the confidence is below the threshold, send an askForUCConfirmation message
3. For each intent amongst the n-best, ask the user if he meant to ask for one of them.
4. The user specifies which is the intent he is searching for by using natural language
   - If the specified intent is amongst the n best, the intent is set and the dialogue goes on
   - Else, the user response is classified again in order to search for the intent

2.3 SLU-level Error Recovery

Like for UC-level error recovery, in SLU-level error recovery there is the need to check if the user requests are correct. This check is done once again by using a threshold.
Even in this case, though, if the object requested by the user gets correctly predicted but it's not in the database, or if it is wrongly predicted but with a high confidence score, there's no error recovery strategy that can possibly work.

2.3.1 SLU n-Best choice

Exactly like in the UC case, the choice of n was made by running the SLU classification for all the utterances in the test file and taking the mean of all the number of predictions above 1% of SLU confidence for each utterance. Once again, a good choice of n in this case was 3.

2.3.2 SLU Threshold

Exactly like the choice for the UC threshold, the SLU acceptance threshold was fixed by running the classification for all the utterances in the test file and taking the smallest value of SLU confidence amongst all the correct predictions.
The SLU-level error recovery strategy works as follows:

1. Check if the SLU confidence is above the threshold
   - If above, return the object
2. If the confidence is below the threshold, send an askForSLUConfirmation message
3. Take the first SLU tag and ask the user if it is correct
   - If the user provides a positive answer, the dialogue goes on
4. If the user provides a negative answer, the system asks the user for the correct object
5. The user provides the correct object via natural language
6. The user's utterance gets classified again in order to fetch the correct object

2.4 Previous considerations on the thresholds

During the development phase, the idea was to set more than one threshold for both UC and SLU. In both cases, at a first stage a rejection threshold was defined. If the utterance provided a UC confidence or SLU confidence under the corresponding threshold, the utterance should have been classified as "bad" and no answer should have been provided.
The confidence area between the rejection threshold and the acceptance one was then considered as an "uncertainty" area, that is, an area in which error recovery was necessary. While under the rejection threshold no error recovery should have been taken into consideration.

But then the idea was to try to always give an answer to the user, so the adopted strategy was to search on Google whenever no match on the DB was found. So the method presented in this method tries to always recover from errors under the (only) acceptance threshold. If it doesn't find any match on the DB, it doesn't completely fail, but it shows some Google results on the topic.

2.5 Results

As expected, introducing error recovery greatly improves the overall number of predictions in some cases. We can see an example below: the same task, "request info on a set of movies" was applied to both the systems with and without error recovery. Table 1 shows the results of the first, while table 2 of the latter.

Table 1: Scenario: without error recovery, Task: request info on a set of movies, Rows: predicted intents, Columns: Correct intents

P \ C	D	M	A	C	RD
director	1	4			
movie		5			
actor		3	2		
country		3		2	
release date		3			2

Table 2: Scenario: with error recovery, Task: request info on a set of movies, Rows: predicted intents, Columns: Correct intents

P \ C	D	M	A	C	RD
director	4	1			
movie		5			
actor		1	4		
country				5	
release date		1			4

Although the tables represent only the number of correct intents predictions (5 utterances per intent), a very similar situation can be obtained when observing the number of concept predictions.

3 Additional features

Here will be listed some of the additional features provided by the system:

- ASR-weighted SLU confidence value: ASR confidence is taken into account for the SLU confidence calculation (by a simple multiplication)
- Barge in: a che ckbox was provided to dynamically turn on and off barge in capability
- (Shallow) Multimodality: as previously said, the system has some multimodal features, like vocal and textual input and output
- Google search: the system always tries to guess a possible response to the user's request

4 Future works

Here will be listed dome of the possible feature works:

- Making the system truly multimodal, by adding the capability of interacting with it in a fully textual way
- Check if the user's question belongs to the movie domain and reject it if it doesn't.
  A good solution would be to reintroduce the UC rejection threshold and consider as out of domain all those questions which intent confidence is below the UC rejection threshold.

A Supplemental Material

- Concept tagger (midterm project on which this one is based)
  https://github.com/A7ocin/LUS_concept_tagger
- Code and README/instructions on how to execute it
  https://github.com/A7ocin/Spoken-Dialogue-System