THE LUNA SPOKEN DIALOGUE SYSTEM: BEYOND UTTERANCE CLASSIFICATION

M. Dinarelli, E.A. Stepanov, S. Varges, G. Riccardi

Department of Information Engineering and Computer Science,
University of Trento, Italy
{dinarelli, stepanov, varges, riccardi}@disi.unitn.it

ABSTRACT

We present a call routing application for complex problem solving tasks. Up to date work on call routing has been mainly dealing with call-type classification. In this paper we take call routing further: Initial call classification is done in parallel with a robust statistical Spoken Language Understanding module. This is followed by a dialogue to elicit further task-relevant details from the user before passing on the call. The dialogue capability also allows us to obtain clarifications of the initial classifier guess. Based on an evaluation, we show that conducting a dialogue significantly improves upon call routing based on call classification alone. We present both subjective and objective evaluation results of the system according to standard metrics on real users.

Index Terms— Spoken Language Understanding, Spoken Dialogue Systems

1. INTRODUCTION

The goal of a call routing application is to determine the type of the call from the (usually first) user utterance and transfer the call to an appropriate destination. Even though at first glance it may seem that a dialogue is not really necessary to accomplish this, it is an important part of such an application since it might not be possible to correctly guess the type of the call. This might happen due to ambiguity of input or failure of the system to understand the user correctly [1]. Call routing has been investigated by [1, 2, 3] among others. Most of the work in call routing has focused on classifiers and their performance. To our knowledge, none of the works in the field report evaluation of the effect of the dialogue on this task.

A call routing application – a Help Desk for hardware and software-related problems – was developed as the Italian Spoken Dialogue System Prototype for the LUNA Project. The goal of the system is to identify the nature of the callers' problems as one belonging to one of the 10 possible scenarios (problem classes). Upon completion of the dialogue, the call is forwarded to the appropriate human operator able to provide further assistance. The system also summarizes call-type and relevant attributes such as brand of hardware. An example dialogue can be seen on Figure 1.

SYSTEM:	Welcome to LUNA. Good day, I am Paola. How may I help you?			
USER:	Eh, Sorry. I have a problem with the printer.			
ASR	And I am sorry a problem with the printer			
SLU		**Concept**	**Value**	**Conf**
		conjugation	and	0.725
		problem	a_problem	0.731
		computer_componentHardware	with_the_printer	0.718
CTC	Label: C1_Printer_Problem; Confidence: 1			
DM	1	Infer subclass_C1_Printer_Problem		
	2	Inferred Class == CTC Label		
	3	CLASS LABEL ONLY		
	4	VERIFY Problem Class		
SYSTEM:	You have a problem with your printer. Do you confirm?			
...	...			
SYSTEM:	Thank you, wait in line.			
	An operator will assist you with your Lexmark printer problem!			

Fig. 1. Example dialogue translated to English

In this paper, we first present the architecture of the Spoken Dialogue System (SDS) in Section 2. Section 3 deals with the Spoken Language Understanding (SLU) component. Section 4 presents the Dialogue Manager (DM). In Section 5 we present both subjective and objective evaluation results. Concluding remarks are presented in Section 6.

2. DIALOGUE SYSTEM ARCHITECTURE

A typical interaction is initiated by a phone call that arrives at a telephony server which routes it to a VXML platform. Since the VXML standard is based on the web infrastructure, a VXML platform can issue HTTP requests that can be served by a web server just like any HTML page. This allows us to organize the processing modules of the dialogue system (SLU, DM, VXML generator) as web services that are invoked by the HTTP request. As a consequence, each system turn of a dialogue is a separate, *stateless* request. Since dialogue is obviously stateful (even more so than conventional web sessions), the state of the conversation needs to be preserved from one turn request to the next. This is done by storing the system state in the database.

The architecture implements a 'fat pipeline': each speech, language and DM module has access to the database for rescoring and modeling (e.g. time series intra and inter dialogues). At the implementation level, this balances a lightweight communication protocol downstream with data flowing laterally towards the database. Further details are described in [4].

3. SPOKEN LANGUAGE UNDERSTANDING

The SLU Module of the SDS consists of two components operating in parallel. Concept tagging is complemented with user call-type classification, and the results of both components are passed to DM to decide the next move.

3.1. Automatic Concept Tagging

The automatic concept tagging module of the prototype is implemented with the re-ranking framework described in [5]. This framework performs SLU in two steps using jointly Generative and Discriminative models.

The first step is to produce a list of SLU hypotheses using a Stochastic Conceptual Language Model. This is the same as in [6] with the difference that we train the language model using the SRILM Toolkit [7] and we then convert it into a SFST. This method allows us to use a wide group of language models, backed-off or interpolated with many kinds of smoothing techniques.

The second step is discriminative re-ranking of the list of n-best SLU hypotheses generated by the Stochastic Finite State Transducer (SFST) module described above. Our discriminative re-ranking is essentially an SVM (i.e. a classifier) trained with pairs of conceptually annotated sentences produced by the SFST. SVMs learn to select which annotation has an error rate lower than the others so that the n-best annotations can be sorted based on their correctness. The kernel used to measure similarity between SLU hypotheses is a Partial Tree Kernel (PTK). This re-ranking kernel, based on the combination of four PTK, has been used successfully in several tasks (see [5] for details on the Partial Tree Kernel and on discriminative re-ranking for SLU) of Natural Language Processing.

Let us consider the following Italian sentence as input, taken from the LUNA corpus along with its English translation: *Ho un problema sulla stampante* (*I have a problem with my printer*). A possible semantic annotation for this sentence is:

null{*ho*} ***PROBLEM-B***{*un*} ***PROBLEM-I***{*problema*}
HARDWARE-B{*sulla*} ***HARDWARE-I***{*stampante*}

where ***PROBLEM*** and ***HARDWARE*** are two domain concepts taken from a purpose-defined ontology. The ontology describes each concept as a class with one or more attributes as well as relations holding between concepts [8]. ***null*** is the label used for words not meaningful for the task. In order to have a one-to-one association between words and concepts, we use *begin (B)* and *inside (I)* markers to segment each concept.

This annotation is automatically performed by a model combining the FST representation of the input sentence with three transducers: (1) transducer mapping words to word categories (POS Tags and morpho-syntactic categories), (2) transducer mapping categories into concepts and (3) the SFST representation of the SCLM mentioned above. The SCLM represents joint probability of word and concept sequences.

The model has been tested on the Italian corpus acquired within the LUNA Project. The corpus is made of human-machine (HM) dialogues acquired with a Wizard-of-Oz approach (WOZ). The results of the SLU model described in this work are expressed in terms of Concept Error Rate (CER). The baseline of the SFST model on the LUNA corpus test set is 23.2% CER and 26.1% CER on attributes and attribute-values extraction, respectively. Applying the re-ranking framework described in this section, results are 18.4% CER and 21.3% CER on attributes and attribute-values extraction. For further details on the LUNA corpus and results of SLU using SFST and the re-ranking framework see [5].

3.2. Call-Type Classification

Since the goal of the system is to classify the call as belonging to one of the ten possible scenarios, SLU concept attribute values and segmentation are also complemented with user goal prediction. These user goals are extracted from the caller responses to the open-ended opening prompt. A BoosTexter [9] based utterance classifier is build to operate on ASR hypotheses in parallel to SLU concept segmentation. The class label provided by the classifier and its confidence are used further for the decision by the DM.

The call-type classifier was trained on first utterances of the LUNA Wizard of Oz corpus. Due to the nature of the task - the classification of documents in the same domain - the most distinguishing features are hardware types such as keyboard, mouse, etc.; thus, unigram model had comparable performance to the bigram and trigram feature models. The best performing unigram model, having 93.8% accuracy on the test set, was chosen to be used for call-type classification within the spoken dialogue system prototype. We attribute the high performance on the test set to the nature of the WOZ data collection and to it being trained and tested on transcriptions. Thus, we expect the performance to be lower in real user interactions involving classification of ASR output, which is known to have lower accuracy than transcription. In fact, the performance of the classifier on ASR outputs of the same test set yields 90.8% accuracy.

4. DIALOGUE MANAGEMENT

Dialogue management follows an Information State Update approach [10]. First, the following information is retrieved from the database:

- ASR recognition results of last user turn;
- confidence and other thresholds;
- SLU concept attribute-value pairs;

- classifier results for the problem class if available (first turn only), which include problem class label and confidence of this label, as mentioned in section 3.2;
- all open questions for the current dialogue from the database;
- application information already provided by the user, including their grounding status ('explicitly-verified' by a clarification question, 'implicitly-accepted' by virtue of being above a heuristically set threshold, and 'under-verification' for ongoing clarification questions);
- user rejections in verification questions and 'noinput' / 'nomatch' events of the entire dialogue;
- 'noinput' and other events are summarized to produce counts that can serve as 'goodness metrics' of the ongoing dialogue.

Given this information, the DM employs a 'dialogue move engine' to determine the system action and response. It is using several sets of forward chaining inference rules: SLU rules match the user utterance to open questions. This may result in the decision to verify the application parameter in question, and the action is verbalized by language generation rules (which are part of the DM in this system). If the parameter is accepted, application dependent task rules determine the next parameter to be acquired, resulting in the generation of an appropriate request. Typical dialogue moves available to the system are those that are needed for the application domain, for example forward looking moves such as 'question-parameter', 'confirm-parameter', and 'request-repeat', and backward looking moves such as 'accept-parameter-implicitly' (by the system) or 'answer-question-parameter' (by the user). The dialogue is initially open to a wide range of user utterances (in response to "How may I help you?"), and becomes more constrained after that.

5. EXPERIMENTS AND RESULTS

The system was tested by 50 volunteer callers (3 calls each), and the collected calls were transcribed and annotated. We have selected a 100 dialogue subset (only dialogues containing all required metrics are included) that was used in assessing the systems performance.

5.1. General Dialogue Statistics

The average duration of the 100 dialogues is 40.29 seconds, with 5.10 turns per dialogue in average. In the scenarios covered by the prototype there is 1 task per dialogue by design (excluding transfer to the operator request, which was never encountered).

The dialogues were categorized with respect to the way the dialogue was ended: T1 – the call was routed with correct attribute values, T2 – the call was routed with incorrect attribute values, and T3 – the call was transferred to the operator. As can be seen in Table 1, successful dialogues (T1) have the least number of turns on average and the shortest dialogue duration. Dialogues in T3 category, on the other hand, have the longest average duration and contain the highest average number of turns. The average turn duration exhibits the reverse tendency compared to average dialogue duration and number of turns. This observation is to be expected since successful dialogues contain fewer turns with more interaction. The general dialogue statistics once again confirm that the length of the dialogue is indicative of its success.

Metric	All	T1	T2	T3
Av. Dial. Dur. (sec)	40.29	36.54	42.93	45.90
Av. Turn. Dur. (sec)	7.90	8.08	7.87	7.40
Av. # of Turns	5.10	4.52	5.45	6.20
Av. # of Tasks	1	1	1	1

Table 1. General dialogue and utterance level metrics

	P	R	F1	TSR
T	0.52	0.46	0.49	0.47
T*	0.57	0.56	0.56	0.57

Table 2. Task Success as Precision (P), Recall (R) and F-Measure (F1); and Task Success Rate (TSR)

5.2. Task Success

The task is considered completed successfully in case it was routed to the appropriate destination, i.e. both problem class and the hardware brand attribute should be correct. However, from the point of view of the user, on a longer term, the task can be considered successful also in case of transfer to the operator. The task success values presented are given for both cases: T – success only refers to call routing with correct attributes; T* – success also includes transfer to the operator.

[4] measure task success as the ratio of completed tasks to tasks requested, such that the completed task is the requested task. Precision (P), Recall (R), F-Measure ($F1$), on the other hand, allows one to measure task success in a way that also takes into account mismatches between requested and completed task types. The resulting overall task success in terms of P, R, & $F1$ and Task Success Rate is presented in Table 2.

Regarding task failure, in the majority of cases (28 out of 43, i.e. 65.12%; not shown in Table 2), a task was not completed successfully (call routings with wrong hardware brand attribute) because the system received a brand name that was not covered by the grammar. Misrecognition was not among the main reasons contributing to the 'failed' task category in case of routing with incorrect attributes or in case of transfer to the operator.

Overall, the task success could be greatly improved by increasing the coverage of the grammars used in the system. However, better ASR is not the only factor affecting successful call routing; classifier performance and dialogue also play an important role. In Section 5.4 we present an evaluation of the classifier in detecting call type as well as the performance increase we gain from dialogue.

Score	T3	T2	T1	Total
5	0.10 (1)	0.27 (12)	0.63 (29)	0.42 (42)
4	0.00 (0)	0.11 (5)	0.22 (10)	0.15 (15)
3	0.10 (1)	0.30 (13)	0.11 (5)	0.19 (19)
2	0.10 (1)	0.18 (8)	0.04 (2)	0.11 (11)
1	0.70 (7)	0.14 (6)	0.00 (0)	0.13 (13)
# of Scores	0.10 (10)	0.44 (44)	0.46 (46)	1.00 (100)
Av. Score	**1.70**	**3.20**	**4.44**	**3.62**

Table 3. Subjective task success assessment normalized by task completion type (counts given in parentheses)

Problem Class	Classifier	Dialogue
C1 Printer	0.89	0.94
C5 Keyboard	0.67	1.00
C9 CD-ROM	0.94	1.00
...	**...**	**...**
Accuracy (all classes)	0.79	0.94
Weighted Av. Acc.	0.79	0.94

Table 4. Classifier vs. Dialogue performance (accuracy) on determining the problem class

5.3. Task Success as Perceived by Caller

The callers also responded to a questionnaire following the dialogue. One of the questions of this questionnaire, "How well do you think the system understood your problem", was intended to measure the callers subjective perception of the task success. The callers scored the system on a 1-5 Likert scale. The results of the mapping of these subjective evaluation scores to the three task completion types described in Section 5.1 are presented in Table 3. As can be seen, call routing with the correct attribute value was judged as better understanding (average score 4.44) compared to the other two task completion types (T2 – 3.20 and T3 – 1.70). Moreover, there is a significant correlation between user judgements and task completion types: Spearman $r(98) = 0.56, p < 0.05$.

5.4. Call-type Classification vs. Dialogue

Since our system performs call-type classification on ASR interpretations of caller utterances, we can assess the performance gained by implementing the whole dialogue system pipeline versus just the call-type classifier. Table 4 provides the call-type classification accuracy for classifier and the dialogue system as a whole. The values for dialogue contain cases for all dialogue completion types in which the system was able to determine the problem class correctly. As can be seen from the table, there is an important improvement in performance on call-type classification when dialogue is used. Even though only 100 dialogues were used, the results are statistically significant: $\chi^2(1, N = 200) = 9.63, p < 0.05$.

To assess the performance we gain by using dialogue even further, we measured the oracle accuracy of the classifier. Dialogue performance on detecting call category (0.94) turned out to be higher than 6-best oracle accuracy of the classifier (0.93). This means that even an oracle that always chooses correctly from the 6 best hypotheses would not be better than our dialogue system.

6. CONCLUSION

In this paper we have presented a call routing application developed as a spoken dialogue system prototype for the LUNA Project. We described a SLU component based on joint generative and discriminative models, complemented with user-goal classification, the results of which are used by rule-based Dialogue Manager for inferences and the decision process. We also presented evaluation results of the system on task success, comparing objective and subjective assessments of the dialogues. We quantified the effect of dialogue on call routing and found a significant improvement compared to routing based on call classification alone.

7. REFERENCES

[1] A. L. Gorin, G. Riccardi, and J. H. Wright, "How may i help you?," *Speech Communication*, vol. 23, pp. 113–127, 1997.

[2] J. Chu-Carroll and B. Carpenter, "Dialogue management in vector-based call routing," in *Proc. of the Annual Meeting of the ACL*, Montreal, 1998.

[3] K. Evanini, D. Suenderman, and R. Pieraccini, "Call classification for automated troubleshooting on large corpora," in *Proc. of ASRU 2007*, Kyoto, Japan, 2007.

[4] S. Varges, G. Riccardi, and S. Quarteroni, "Persistent information state in a data-centric architecture," in *Proc. of SIGDial 2008*, Columbus, USA, 2008.

[5] M. Dinarelli, A. Moschitti, and G. Riccardi, "Re-ranking models for spoken language understanding," in *Proc. of EACL2009*, Athens, Greece, 2009.

[6] C. Raymond and G. Riccardi, "Generative and discriminative algorithms for spoken language understanding," in *Proc. of Interspeech 2007*, Antwerp, Belgium, 2007.

[7] A. Stolke, "Srilm: an extensible language modeling toolkit," in *Proc. of SLP2002*, Denver, USA, 2002.

[8] S. Quarteroni, G. Riccardi, and M. Dinarelli, "What's in an ontology for spoken language understanding," in *Proc. of Interspeech 2009*, Brighton, U.K., 2009.

[9] R.E. Schapire and Y. Singer, "Boostexter: A boosting-based system for text categorization," *Machine Learning*, vol. 39, no. 2/3, pp. 135–168, 2000.

[10] S. Larsson and D. Traum, "Information state and dialogue management in the trindi dialogue move engine toolkit," *Natural Language Engineering*, vol. 6, no. 3-4, pp. 323–340, 2000.