Robust language analysis and generation for spoken dialogue systems

Sebastian Varges and **Matthew Purver**
Center for the Study of Language and Information
Stanford University
Stanford, CA 94305, USA
{varges, mpurver}@stanford.edu

Abstract. Spoken dialogue systems have to deal with imperfect speech recognition. We describe how we address the resulting robustness challenges both on the interpretation side (via combining complementary parsing and classification techniques) and on the generation side (via robustness to ill-formed generation input, and use of clarification questions). We describe empirical results obtained from a user study involving 20 subjects.

1 Introduction

This paper explores techniques that address robustness challenges in an implemented spoken dialogue system. As we will show, decisions about robustness issues on the interpretation side impact the robustness challenges faced by the generator.

The dialogue system uses an domain-independent architecture that allows one to tailor it to different tasks. One of its tasks, the one we are focussing on in this paper, is to collaboratively compose a search query with the user in information seeking dialogues, for example in restaurant selection tasks. In other words, logical forms are constructed interactively, not as a result of one-shot user queries.

In a very wide sense, our dialogue strategies themselves lead to increased 'robustness' in terms of successful database queries (i.e. queries with non-empty result sets): rather than just reporting that the query returned no matches, our system actively relaxes constraints and encourages the user to refine the query further if the result set is large. However, in this paper we concentrate on the more low-level details of how to process speech recognition (ASR) input and how to robustly generate appropriate system responses.

In the next section, we give a system overview. We describe interpretation and generation components in sections 3 and 4. In section 5 we discuss robustness issues and section 6 presents evaluation results from a user study. This is followed by conclusions.

2 The dialogue system

The dialogue system [13] employs the following architecture: The output of a speech recognizer (Nuance) is analyzed by a context-independent natural language understanding (NLU) module; the output of this module is then disambiguated and instantiated in context by the dialogue manager (DM). The approach to dialogue management follows [6, 7] in its use of a rich tree-based model of context, together with a 'plug-and-play' multi-device architecture which allows the underlying devices to specify their dialogue interfaces via 'dialogue move scripts'. These scripts declaratively license the possible instantiations of dialogue moves and their integration into the 'dialogue move tree'. In order to tailor the system to new domains, only the scripts need to be adapted, not the underlying machinery implemented in Java. The dialogue system is fully implemented and has been used in restaurant selection and MP3 player tasks. There are 41 task-independent, generic dialogue move rules, 52 restaurant selection rules and 89 MP3 player rules.

Query constraints can be associated with dialogue moves if the NLU output matches particular input patterns specified in the scripts. For example, in the Restaurant domain, a request such as "I want to find an inexpensive Japanese restaurant that takes reservations" results in the semantic frame below:

```
(1) system:Category = restaurant:Restaurant
    restaurant:PriceLevel = 0-10
    restaurant:Cuisine = restaurant:japanese
    restaurant:Reservations = yes
```

This frame is used to construct a database query. If the query returns no results, various constraint modification strategies such as constraint relaxation or removal can be employed. For example, 'Japanese food' can be relaxed to 'Asian food' since cuisine types are hierarchically organized.

3 Interpretation

3.1 Combining complementary hypotheses

The approach to robust interpretation centres around the use of multiple independent interpretation methods, both shallow and deep, and open- and closed-domain. The NLU module contains both a broad-coverage statistical dependency parser and a domain-specific slot/value-based semantic classifier. The semantic classifier operates under a closed-world assumption and must be trained on in-domain data; its slots correspond to the possible constraint types in the current device's database (e.g. price level and cuisine type, as shown above), and the possible values correspond to the available values in the relevant ontology. The classification method may depend on the nature of the slot and its possible value space (the current implementation uses a maximum-entropy approach [14] for some, and simpler pattern-matching methods for others), but is inherently robust to the exact *form* of the input, and thus to unexpected surface forms and to ASR errors relating to function words (e.g. (1a) below). However, it has the disadvantage of being closed-domain, and is thus not robust

to misrecognized or unexpected (either truly out-of-domain, or just unseen in training data) semantically potent words. In contrast, the dependency parser (a version of [12]) has no knowledge of the ontology, but produces Penn Treebank-style syntactic structures; it is thus often less accurate, and is sensitive to surface form, but is relatively robust to unexpected or misrecognized nouns or verbs, as in (1b):

(1) a. In-domain constraint, unexpected form:
"kind of uh kind of fill like Thai food"
b. Out-of-domain constraint, parsable form:
"I'm looking for a Montenegrin restaurant"

This has some similarities to [3]'s use of dual ASR, where a general statistical language model version is used to detect words or phrases which the main domain-grammar-based ASR and NLU cannot treat. However, as their NLU is closed-world, that approach is essentially limited to telling the user why input is rejected; whereas we can pass both types of NLU output on to the DM, either for successful resolution (if in-domain but unseen) or generation of targeted clarification or other helpful responses (e.g. via constraint relaxation).

Our dialogue management strategy allows active devices, and their active nodes in the dialogue move tree (possible antecedent moves in context) to produce move hypotheses from either type of NLU output alone, or from combinations of the two.[1] Each hypothesis is given an overall score, based on the confidence assigned by the particular NLU agent as well as a combination of pragmatic contextual factors such as the number of phrases which can be resolved as plausible constraints (see [2] for similar inclusion of pragmatic factors, although within a closed-world ASR/NLU approach).

3.2 Combined confidence scoring

The overall score used to choose the most likely move hypothesis in context is also used to drive confirmation/clarification strategies. The highest-scoring alternative can only be accepted without question if its score exceeds a given threshold; below this, confirmation or clarification is required, but the exact behaviour is driven by two further thresholds. In the upper range, the move is accepted but implicitly confirmed in the next system move; in the middle range, an explicit confirmation question is asked; and in the lowest range an non-understanding message is generated, although this may include a topic-specific help suggestion if a confident-enough semantic slot value is present. We therefore have a similar strategy to [8], but incorporate more levels of information to give a more useful measure of confidence: a high ASR confidence can still lead to clarification if the only semantic interpretation is highly implausible; and conversely, high contextual plausibility can lead to acceptance even with low ASR confidence (although perhaps with implicit confirmation).

We are currently investigating the use of the scores at individual levels (ASR, NLU or context) to drive clarification strategies and determine the best question to ask in a particular situation (one which is most likely to produce information at the level in which the system has least confidence); together with the use of relative move hypothesis scores as well as absolute ones.

3.3 Hypothesis & confirmation

Importantly, our rich representation of context means that we can maintain a representation of the best hypothesis (and its connection to context) while asking a related confirmation or clarification question. Given our tree-based representation, the hypothesised node is not attached to the tree itself, but held embedded within a confirmation node. This allows an ensuing answer to this question to have a full range of effects: a positive confirmation causes the hypothesised node to be attached to its antecedent and have its normal effects; a negative answer can prevent its attachment; and importantly, answers providing new or contradictory information can have their constraints integrated into the hypothesis. This allows sequences such as "Are you looking for a Thai restaurant? / No, a Chinese one", or "Are you looking for a Thai restaurant? / Yes, one that takes credit cards."

[1] For example, a script may define a particular move type as being a possible hypothesis if the parser output matches a particular (possibly underspecified) syntactic form, while the semantic classifier assigns a reasonably confident value to a given slot.

4 Natural Language Generation (NLG)

Sentences s1-s4 in table 1 show the main dialogue strategies for presenting the results of database queries. If the number of items returned for the database query is large, the system suggests further (unused) constraints (sentence s4). The task of the generator is to produce these verbalizations given dialogue strategy, constraints and further discourse context.

Overgeneration and ranking approaches to NLG have become increasingly popular [5, 11]. We apply them to dialogue processing and perform mild overgeneration of candidate moves, followed by ranking. The highest-ranked candidate is selected for output.

4.1 Chart generation

We follow the bottom-up chart generation approach [4] for production systems described in [10]. The rule-based core of the generator is a set of productions. Productions map individual database constraints to phrases such as "open for lunch", "within 3 miles" and "a formal dress code", and recursively combine them into NPs. This includes the use of coordination to produce "restaurants with a 5-star rating and a formal dress code", for example. The NPs are integrated into sentence templates, several of which can be combined to form an output candidate turn.

The selection of which sentence template to use is determined by the dialogue move scripts. Typically, a move-realizing production produces several alternative sentences. On the other hand, the NP generation rules realize constraints regardless of the specific dialogue move at hand. This allows us to also use them for clarification questions; all that is required is a new set of sentence templates, for example "Are you looking for [NP]" for the middle range of the confidence score (see section 3.2) and "I'm sorry but I did not understand what you mean by [String]" for the lower range. Note that in the latter case we need to repeat the input string since we do not have a detailed analysis. However, a confident-enough semantic slot value allows us to generate a possible help suggestion "If you're looking for a particular kind of food, try saying something like 'I want Indian food'." (Also note that the upper range of the confidence score is reflected by examples s1-s4 in table 1).

We currently use 102 productions overall in the restaurant and MP3 domains, 38 of them to generate NPs that realize 19 possible input constraints (in both domains).

4.2 Ranking: alignment & variation

Alignment Alignment is a key to successful natural language dialogue [1]. We perform alignment of system utterances with user utterances by computing an ngram-based overlap score. For example, a user utterance "I want to find a Chinese restaurant" is presented by

	$\|result\|$	example realization	f_{exp}
s1	0	I'm sorry but I found no restaurants on Mayfield Road that serve Mediterranean food .	0
s2	small: $> 0, < t_1$	There are 2 cheap Thai restaurants in Lincoln in my database : Thai Mee Choke and Noodle House .	61
s3	medium: $>= t_1, < t_2$	I found 9 restaurants with a two star rating and a formal dress code that are open for dinner and serve French food . Here are the first ones :	212
s4	large: $>= t_2$	I found 258 restaurants on Page Mill Road, for example Maya Restaurant , Green Frog and Pho Hoa Restaurant . Would you like to try searching by cuisine ?	300
s5	(any)	I found 18 items .	2

Table 1. System responses for database queries with different result set sizes. s5 is a default responses produced by a template.

the bag-of-words {'I', 'want', 'to', 'find', ...} and the bag-of-bigrams {'I want', 'want to', 'to find', ...}. Words are lemmatized and proper nouns of example items removed from the utterances. We compute the overlap with system utterances represented in the same way and combine the unigram and bigram match scores.

Variation We also use a variation score to 'cycle' over sentence-level paraphrases. Alternative candidates for realizing a certain input move are given a unique alternation ('alt') number in increasing order. For example, for the simple move `continuation_query` we may assign the following alt values: "Do you want more?" (alt=1), "Do you want me to continue?" (alt=2), and "Shall I continue?" (alt=3). The system cycles over these alternatives in turn. Once we reach alt=3, it starts over from alt=1. The actual alt 'score' is inversely related to recency and normalized to [0...1].

Score combination The final candidate score is a linear combination of alignment and variation scores:

$$score_{final} = \lambda_1 \cdot align_{uni,bi} \quad +(1-\lambda_1) \cdot variation \qquad (1)$$

$$align_{uni,bi} = \lambda_2 \cdot align_{uni} \quad +(1-\lambda_2) \cdot align_{bi} \qquad (2)$$

where $\lambda_1, \lambda_2 \in \{0...1\}$. A high value of λ_1 places more emphasis on alignment, a low value yields candidates that are more different from previously chosen ones. In our experience, alignment should be given a higher weight than variation, and, within alignment, bigrams should be weighted higher than unigrams, i.e. $\lambda_1 > 0.5$ and $\lambda_2 < 0.5$. Deriving weights empirically from corpus data is an avenue for future research.

5 Robustness issues

Robustness challenges for interpretation modules are generally acknowledged. Generation is often seen as a downstream activity in dialogue processing (which assumes that the user utterance is the starting point of processing), and thus not often considered as a focal point when considering system robustness. However, we find that speech recognition errors filter through to the generator, and that further errors – themselves possibly triggered by speech recognition errors – may occur before generation and be passed on. In our practical experience (see section 6) we find the following sources of errors:

1. Inaccurate parsing and semantic classification. The parser may produce a structure which looks like a query with a cuisine-name argument (leading to creation of a database constraint based on that argument), although the words forming that argument are nothing to do with cuisine type.
2. Misrecognized or out-of-domain words. This may lead to unexpected constraints, sometimes intended by the user and sometimes not, e.g. "Montenegrin restaurant" in example (1b).
3. The generation grammar may be incomplete, i.e. the grammar may not be able to generate turns that express all input constraints.
4. The generation input may be incomplete, omitting crucial information needed to construct complete sentences. For example, the category of the retrieved database items – such as 'restaurant' or 'brewpub' – may be missing. In that case, the grammar cannot generate a head noun to which it can attach realizations of other constraints (e.g. a prepositional phrase "on Bower Street").

It seems that the first and second problems, which we may call 'erroneous constraints', can only be solved partially, by asking appropriate clarification questions and/or informing the user of the queries performed. This requires clarification/confirmation to be driven by some pragmatic information (see section 3.2), and generation to be robust to out-of-domain expressions (4.1). In some cases, DM-internal modules can apply additional sanity checks to block erroneous constraints. For example, the database may know that a certain number is an impossible price range for a restaurant. This in turn may lead to 'incomplete' generation output. Thus, we generally cannot require candidates to be complete, i.e. to express all constraints. As a consequence, we regard all generated sentences as candidates and factor completeness into the selection/ranking process [9] (this also addresses issue 3 above). Combined with the robustness of bottom-up processing known from parsing, this allows the generator to generate maximally complete output.

We address issue 4 above by using rules that add additional category constraints as defaults in case these are missing. From the existing constraints we can infer that we are dealing with the restaurant domain rather than the MP3 domain. However, we will not be able to make fine-grained distinctions such as asserting 'brewpub' rather than 'restaurant', for example.

As a further technique to increase robustness, we use a simple template generator as a fall-back if NLG fails (see s5 in table 1).

Considering the fact that the domain ontology and database schema are known in advance, it is tempting to make a closed world assumption in the generator (which could also help system development and testing). However, this seems too restrictive: assume, for example, that the user has asked for Montenegrin food, which is an unknown cuisine type, and that the statistical parser combined with the parse-matching patterns in the DM has labeled this correctly. The content optimization module will remove this constraint since there is no Montenegrin restaurant in the database. If we now want to generate "I did not find any restaurants that serve Montenegrin food ...", we do need to be able to use generation input that uses unseen

attribute-value pairs. The price one has to pay for this increased robustness and flexibility is, of course, potentially bad output if NLU mislabels input words. More precisely, we find that if any one of the interpretation modules makes an open-world assumption, the generator has to do as well, at least as long as we want to verbalize the output of that module.

6 User study

Each of 20 subjects in a restaurant selection task was given 9 scenario descriptions involving 3 constraints (see [13]). Subjects were instructed to use their own words to find a fitting restaurant. The back-end database contained 2500 restaurants containing the 13 attributes/constraints for each restaurant, for example cuisine type, city and street names, service, rating, price, open hours, dress code etc.

Overall, 180 tasks were performed involving 1144 user turns and 1818 system turns; task completion rate was 94%. Two factors contributing to the higher number of system turns are a) some system turns are counted as two turns if they contained several dialogue moves, and b) restaurants in longer enumerations of result items are counted as individual turns. On average, user utterances are significantly shorter than system utterances (4.9 words, standard deviation $\sigma = 3.82$ vs 15.4 words, $\sigma = 13.53$). This is a result of the longer 'constraint summaries' produced by the generator. These descriptions of what the system understood and what actions it performed based in its understanding are crucial for providing suitable feedback to the user. The high standard deviation of the system utterances can be explained by the listing of individual result items.

On the interpretation side, semantic accuracy (query constraints correctly produced) gave an f-score of 82.2%. Approximately 38% of utterances could only be treated by the parser, and 25% only by the classifier.[2] Post-processing of a subset of system logs shows that the use of combined scoring for best interpretation gives a reduction in error of 30-45% over using parser or classifier alone (perhaps unsurprisingly), but more interestingly an error reduction of 10-15% over a strategy of using both but making the choice on NLU n-best list position alone.

The generator asserted missing constraints to incomplete generation input in 30 cases out of 579 inputs that were comprised of database constraints (in contrast to realizing other dialogue moves). The last column of table 1 shows the usage frequencies of the basic dialogue strategies which depend on the sizes of result sets returned from the database. There are only two cases in which the system had to fall back to the default template generator. Furthermore, we find that (almost) all generation output is complete. Thus, while the relaxation of the completeness requirement is very useful for system development, the grammar is capable of expressing the constraints and constraint combinations used in the application domain. However, in more open and less well-defined domains, we expect a relaxed completeness requirement to play a greater rôle.

7 Conclusions

Robustness challenges, in particular due to speech recognition errors, require a set of techniques that address the problem at different levels. As this paper has shown, no single technique seems to be able to 'solve' all of these. In our implemented system, the used techniques are, at different stages of dialogue processing:

[2] Note that many parser-only moves were those which the classifier was not designed to handle, e.g. yes/no answers.

Interpretation: We find that combining complementary NLU techniques can give better performance than using either one alone. We also allow confidence factors at multiple levels of interpretation (including context) to assist in improving choice of hypothesis, and in driving clarification and confirmation strategies.

Generation: Most work on generation for practical dialogue systems makes use of generation components that work toward a single output (often simple template-based generation). We find that language generation faces robustness challenges similar to those of interpretation, and that similar techniques can be applied: ranking of alternatives, bottom-up chart-based processing, and relaxation of completeness constraints (corresponding to partial parsing). This can be combined with sanity checks of the input to the generator (e.g. ranges of numerical values), and insertion of defaults for missing input semantics. We find that the choices made in the interpretation modules affect language generation, for example whether or not a closed-world assumption can be made.

ACKNOWLEDGEMENTS

This work is supported by the US government's NIST Advanced Technology Program; partners are CSLI, Robert Bosch Corporation, VW America, and SRI International. We thank the many people involved in system design, development and evaluation, in particular those who performed the evaluation (VW/Bosch) and developed the NLU and content optimizer modules (Bosch) – see [13].

REFERENCES

[1] C. Brockmann, A. Isard, J. Oberlander, and M. White, 'Modelling alignment for affective dialogue', in *Proc. of the UM'05 Workshop on Adapting the Interaction Style to Affective Factors.*, (2005).

[2] M. Gabsdil and O. Lemon, 'Combining acoustic and pragmatic features to predict recognition performance in spoken dialogue systems', in *Proc. ACL-04*, pp. 344–351, Barcelona, (2004).

[3] B. A. Hockey, O. Lemon, E. Campana, L. Hiatt, G. Aist, J. Hieronymus, A. Gruenstein, and J. Dowding, 'Targeted help for spoken dialogue systems', in *Proc. EACL'03: 10th Conference of the European Chapter of the Association for Computational Linguistics*, Budapest, (2003).

[4] M. Kay, 'Chart generation', in *Proc. ACL-96*, pp. 200–204, (1996).

[5] I. Langkilde, 'An empirical verification of coverage and correctness for a general-purpose sentence generator', in *Proc. INLG-02 (International Natural Language Generation Conference)*, (2002).

[6] O. Lemon and A. Gruenstein, 'Multithreaded context for robust conversational interfaces', *ACM Transactions on Computer-Human Interaction*, **11**(3), (2004).

[7] D. Mirkovic and L. Cavedon, 'Practical plug-and-play dialogue management', in *Proc. Annual Meeting of the Pacific Association of Computational Linguistics (PACLING)*, Tokyo, (2005).

[8] R. San-Segundo, J. M. Montero, J. Ferreiros, R. Córdoba, and J. M. Pardo, 'Designing confirmation mechanisms and error recover techniques in a railway information system for spanish', in *Proc. 2nd SIGdial Workshop on Discourse and Dialogue*, Aalborg, (2001).

[9] S. Varges, 'Fluency and completeness in instance-based natural language generation', in *Proc. COLING-02*, (2002).

[10] S. Varges, 'Chart generation using production systems', in *Proc. 10th European Workshop On Natural Language Generation*, (2005).

[11] S. Varges and C. Mellish, 'Instance-based natural language generation', in *Proc. NAACL-01*, (2001).

[12] F. Weng, N. Jin, J. Meng, and Y. Zhu, 'A Novel Probabilistic Model for Link Unification Grammar', in *7th International Workshop on Parsing Technologies*, Beijing, (2001).

[13] F. Weng, S. Varges, B. Raghunathan, F. Ratiu, H. Pon-Barry, B. Lathrop, Q. Zhang, T. Scheideck, H. Bratt, K. Xu, M. Purver, R. Mishra, M. Raya, S. Peters, Y. Meng, L. Cavedon, and L. Shriberg, 'CHAT: A conversational helper for automotive tasks'. In submission, 2006.

[14] Y. Zhou, F. Weng, L. Wu, and H. Schmidt, 'A fast algorithm for feature selection in conditional maximum entropy modeling', in *Proc. Empirical Methods in Natural Language Processing*, Sapporo, (2003).