Response-Based Confidence Annotation for Spoken Dialogue Systems

Alexander Gruenstein
Spoken Language Systems Group
M.I.T. Computer Science and Artificial Intelligence Laboratory
32 Vassar St, Cambridge, MA 02139 USA
alexgru@csail.mit.edu

Abstract

Spoken and multimodal dialogue systems typically make use of confidence scores to choose among (or reject) a speech recognizer's N-best hypotheses for a particular utterance. We argue that it is beneficial to instead choose among a list of candidate system *responses*. We propose a novel method in which a confidence score for each response is derived from a classifier trained on acoustic and lexical features emitted by the recognizer, as well as features culled from the generation of the candidate response itself. Our response-based method yields statistically significant improvements in F-measure over a baseline in which hypotheses are chosen based on recognition confidence scores only.

1 Introduction

The fundamental task for any spoken dialogue system is to determine how to respond at any given time to a user's utterance. The challenge of understanding and correctly responding to a user's natural language utterance is formidable even when the words have been perfectly transcribed. However, dialogue system designers face a greater challenge because the speech recognition hypotheses which serve as input to the natural language understanding components of a system are often quite errorful; indeed, it is not uncommon to find word error rates of 20-30% for many dialogue systems under development in research labs. Such high error rates often arise due to the use of out-of-vocabulary words, noise, and the increasingly large vocabularies of more capable systems which try to allow for greater naturalness and variation in user input.

Traditionally, dialogue systems have relied on confidence scores assigned by the speech recognizer to detect speech recognition errors. In a typical setup, the dialogue system will choose to either accept (that is, attempt to understand and respond to) or reject (that is, respond to the user with an indication of non-understanding) an utterance by thresholding this confidence score.

Stating the problem in terms of choosing whether or not to accept a particular utterance for processing, however, misses the larger picture. From the user's perspective, what is truly important is whether or not the system's response to the utterance is correct. Sometimes, an errorful recognition hypothesis may result in a correct response if, for example, proper names are correctly recognized; conversely, a near-perfect hypothesis may evoke an incorrect response. In light of this, the problem at hand is better formulated as one of assigning a confidence score to a system's candidate response which reflects the probability that the response is an acceptable one. If the system can't formulate a response in which it has high confidence, then it should clarify, indicate non-understanding, and/or provide appropriate help.

In this paper, we present a method for assigning confidence scores to candidate system responses by making use not only of features obtained from the speech recognizer, but also of features culled from the process of generating a candidate system response, and derived from the distribution of candidate responses themselves. We first compile a list of unique candidate system responses by processing

Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue, pages 11–20,
Columbus, June 2008.

each hypothesis on the recognizer's N-best list. We then train a Support Vector Machine (SVM) to identify acceptable responses. When given a novel utterance, candidate responses are ranked with scores output from the SVM. Based on the scores, the system can then either respond with the highest-scoring candidate, or reject all of the candidate responses and respond by indicating non-understanding.

Part of the motivation for focusing our efforts on selecting a system response, rather than a recognition hypothesis, can be demonstrated by counting the number of unique responses which can be derived from an N-best list. Figure 1 plots the mean number of unique system responses, parses, and recognition hypotheses given a particular maximum N-best list length; it was generated using the data described in section 3. Generally, we observe that about half as many unique parses are generated as recognition hypotheses, and then half again as many unique responses. Since many hypotheses evoke the same response, there is no value in discriminating among these hypotheses. Instead, we should aim to gain information about the quality of a response by pooling knowledge gleaned from each hypothesis evoking that response.

We expect a similar trend of multiple hypotheses mapping to a single parse in any dialogue system where parses contain a mixture of key syntactic and semantic structure—as is the case here—or where they contain only semantic information (*e.g.*, slot/value pairs). Parsers which retain more syntactic structure would likely generate more unique parses, however many of these parses would probably map to the same system response since a response doesn't typically hinge on every syntactic detail of an input utterance.

The remainder of our discussion proceeds as follows. In section 2 we place the method presented here in context in relation to other research. In section 3, we describe the City Browser multimodal dialogue system, and the process used to collect data from users' interactions with the system. We then turn to our techniques for annotating the data in section 4 and describe the features which are extracted from the labeled data in section 5. Finally, we demonstrate how to build a classifier to rank candidate system responses in section 6, which we evaluate in section 7.

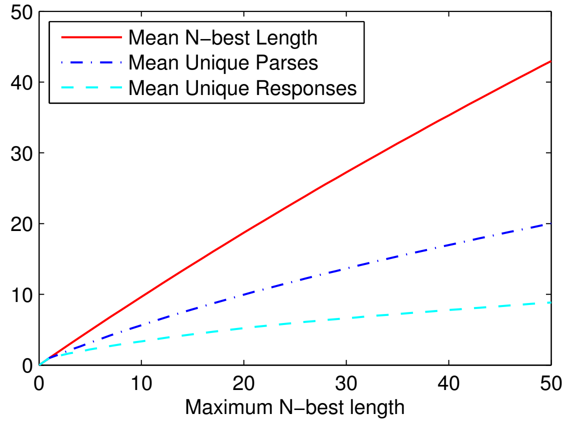

Figure 1: The mean N-best recognition hypothesis list length, mean number of unique parses derived from the N-best list of recognition hypotheses, and mean number of unique system responses derived from those parses, given a maximum recognition N-best list length.

2 Related Work

There has been much research into deriving utterance-level confidence scores based on features derived from the process of speech recognition. The baseline utterance-level confidence module we make use of in this paper was introduced in (Hazen et al., 2002); we use a subset of the recognizer-derived features used by this module. In it, confidence scores are derived by training a linear projection model to differentiate utterances with high word error rates. The utterance-level confidence scores are used to decide whether or not the entire utterance should be accepted or rejected, while the decision as to how to respond is left out of the classification process. Of course, most other recognizers make use of utterance or hypothesis level confidence scores as well; see, for example (San-Segundo et al., 2000; Chase, 1997).

(Litman et al., 2000) demonstrate the additional use of prosodic features in deriving confidence scores, and transition the problem from one of word error rate to one involving concept error rate, which is more appropriate in the context of spoken dialogue systems. However, they consider only the top recognition hypothesis.

Our work has been heavily influenced by (Gabsdil and Lemon, 2004), (Bohus and Rudnicky, 2002), (Walker et al., 2000), and (Chotimongkol and Rudnicky, 2001) all of which demonstrate the utility of training a classifier with features derived from the natural language and dialogue management components of a spoken dialogue system to better predict the quality of speech recognition results. The work described in (Gabsdil and Lemon, 2004) is especially relevant, because, as in our experiments, the dialogue system of interest provides for map-based multimodal dialogue. Indeed, we view the experiments presented here as extending and validating the techniques developed by Gabsdil and Lemon. Our work is novel, however, in that we reframe the problem as choosing among system responses, rather than among recognizer hypotheses. By recasting the problem in these terms, we are able to integrate information from all recognition hypotheses which contribute to a single response, and to extract distributional features from the set of candidate responses. Another key difference is that our method produces confidence scores for the candidate responses themselves, while the cited methods produce a decision as to whether an utterance, or a particular recognition hypothesis, should be accepted, rejected, or (in some cases), ignored by the dialogue system.

In addition, because of the small size of the dataset used in (Gabsdil and Lemon, 2004), the authors were limited to testing their approach with leave-one-out cross validation, which means that, when testing a particular user's utterance, other utterances from the same user also contributed to the training set. Their method also does not provide for optimizing a particular metric—such as F-measure—although, it does solve a more difficult 3-class decision problem. Finally, another key difference is that we make use of an n-gram language model with a large vocabulary of proper names, whereas theirs is a context-free grammar with a smaller vocabulary.

(Niemann et al., 2005) create a dialogue system architecture in which uncertainty is propagated across each layer of processing through the use of probabilities, eventually leading to posterior probabilities being assigned to candidate utterance interpretations. Unlike our system, in which we train a single classifier using arbitrary features derived from each stage of processing, each component (recognizer, parser, *etc*) is trained separately and must be capable of assigning conditional probabilities to its output given its input. The method hinges on probabilistic inference, yet it is often problematic to map a speech recognizer's score to a probability as their approach requires. In addition, the method is evaluated only in a toy domain, using a few sample utterances.

3 Experimental Data

The data used for the experiments which follow were collected from user interactions with City Browser, a web-based, multimodal dialogue system. A thorough description of the architecture and capabilities can be found in (Gruenstein et al., 2006; Gruenstein and Seneff, 2007). Briefly, the version of City Browser used for the experiments in this paper allows users to access information about restaurants, museums, and subway stations by navigating to a web page on their own computers. They can also locate addresses on the map, and obtain driving directions. Users can interact with City Browser's map-based graphical user interface by clicking and drawing; and they can speak with it by talking into their computer microphone and listening to a response from their speakers. Speech recognition is performed via the SUMMIT recognizer, using a trigram language model with dynamically updatable classes for proper nouns such as city, street, and restaurant names—see (Chung et al., 2004) for a description of this capability. Speech recognition results were parsed by the TINA parser (Seneff, 1992) using a hand-crafted grammar. A discourse module (Filisko and Seneff, 2003) then integrates contextual knowledge. The fully formed request is sent to the dialogue manager, which attempts to craft an appropriate system response—both in terms of a verbal and graphical response. The GENESIS system (Seneff, 2002) uses hand-crafted generation rules to produce a natural language string, which is sent to an off-the-shelf text-to-speech synthesizer. Finally, the user hears the response, and the graphical user interface is updated to show, for example, a set of search results on the map.

3.1 Data Collection

The set of data used in this paper was collected as part of a controlled experiment in which users worked through a set of scenarios by accessing the City Browser web page from their own computers, whenever and from wherever they liked. Interested readers may refer to (Gruenstein and Seneff, 2007) for more information on the experimental setup, as well as for an initial analysis of a subset of the data used here. Users completed a warmup scenario in which they were simply told to utter "Hello City Browser" to ensure that their audio setup and web browser were working properly. They then worked through ten scenarios presented sequentially, followed by time for "free play" in which they could use the system however they pleased.

As users interact with City Browser, logs are made recording their interactions. In addition to recording each utterance, every time a user clicks or draws with the mouse, these actions are recorded and time-stamped. The outputs of the various stages of natural language processing are also logged, so that the "dialogue state" of the system is tracked. This means that, associated with each utterance in the dataset is, among other things, the following information:

- a recording of the utterance;
- the current dialogue state, which includes information such as recently referred to entities for anaphora resolution;
- the state of the GUI, including: the current position and bounds of the map, any points of interest (POIs) displayed on the map, *etc.*;
- the contents of any dynamically updatable language model classes; and
- time-stamped clicks, gestures, and other user interface interaction performed by the user before and during speech.

The utterances of 38 users who attempted most or all of the scenarios were transcribed, providing 1,912 utterances used in this study. The utterances were drawn only from the 10 "real" scenarios; utterances from the initial warmup and final free play tasks were discarded. In addition, a small number of utterances were eliminated because logging glitches made it impossible to accurately recover the dialogue system's state at the time of the utterance.

The class n-gram language model used for data collection has a vocabulary of approximately 1,200 words, plus about 25,000 proper nouns.

4 Data Annotation

Given the information associated with each utterance in the dataset, it is possible to "replay" an utterance to the dialogue system and obtain the same response—both the spoken response and any updates made to the GUI—which was originally provided to the user in response to the utterance. In particular, we can replicate the *reply_frame* which is passed to GENESIS in order to produce a natural language response; and we can replicate the *gui_reply_frame* which is sent to the GUI so that it can be properly updated (*e.g.*, to show the results of a search on the map).

The ability to replicate the system's response to each utterance also gives us the flexibility to try out alternative inputs to the dialogue system, given the dialogue state at the time of the utterance. So, in addition to transcribing each utterance, we also passed each transcript through the dialogue system, yielding a system response. In the experiments that follow, we considered the system's response to the transcribed utterance to be the *correct* response for that utterance. It should be noted that in some cases, even given the transcript, the dialogue system may *reject* and respond by signally non-understanding—if, for example, the utterance can't be parsed. In these cases, we take the response *reject* to be the correct response.

We note that labeling the data in this fashion has limitations. Most importantly, the system may respond inappropriately even to a perfectly transcribed utterance. Such responses, given our labeling methodology, would incorrectly be labeled as *correct*. In addition, sometimes it may be the case that there are actually several acceptable responses to a particular utterances.

5 Feature Extraction

For each utterance, our goal is to produce a set of candidate system responses, where each response is also associated with a vector of feature values to be used to classify it as *acceptable* or *unacceptable*. Responses are labeled as *acceptable* if they match the system response produced from the transcription, and as *unacceptable* otherwise.

We start with the N-best list output by the speech recognizer. For each hypothesis, we extract a set

Recognition			Distributional	Response
(a) Best across hyps:	**(b) Drop:**	**(c) Other:**	percent_top_3	response_type
total_score_per_word	total_drop	mean_words	percent_top_5	num_found
acoustic_score_per_bound	acoustic_drop	top_rank	percent_top_10	POI_type
lexical_score_per_word	lexical_drop	n-best_length	percent_nbest	is_subset
			top_response_type	parse_status
			response_rank	geographical_filter
			num_distinct	

Table 1: Features used to train the acceptability classifier. Nine features are derived from the recognizer; seven have to do with the distribution of responses; and six come from the process of generating the candidate response.

of acoustic, lexical, and total scores from the recognizer. These scores are easily obtained, as they comprise a subset of the features used to train the recognizer's existing confidence module; see (Hazen et al., 2002). The features used are shown in Table 1a.

We then map each hypothesis to a candidate system response, by running it through the dialogue system given the original dialogue state. From these outputs, we collect a list of *unique* responses, which is typically shorter than the recognizer's N-best list, as multiple hypotheses typically map to the same response.

We now derive a set of features for each unique response. First, each response inherits the best value for each recognizer score associated with a hypothesis which evoked that response (see Table 1a). In addition, the drop in score between the response's score for each recognition feature and the top value occurring in the N-best list is used as a feature (see Table 1b). Finally, the rank of the highest hypothesis on the N-best list which evoked the response, the mean number of words per hypothesis evoking the responses, and the length of the recognizer's N-best list are used as features (see Table 1c).

Distributional features are also generated based on the distribution of hypotheses on the N-best list which evoked the same response. The percent of times a particular response is evoked by the top 3, top 5, top 10, and by all hypotheses on the N-best list are used as features. Features are generated, as well, based on the distribution of responses on the list of unique responses. These features are: the initial ranking of this response on the list, the number of distinct responses on the list, and the type of response that was evoked by the top hypothesis on the recognizer N-best list.

Finally, features derived from the response itself, and natural language processing performed to derive that response, are also calculated. The high-level type of the response, as well as the type and number of any POIs returned by a database query are used as features if they exist, as is a boolean indicator as to whether or not these results are a subset of the results currently shown on the display. If any sort of "geographical filter", such as an address or circled region, is used to constrain the search, then the type of this filter is also used as a feature. Finally, the "best" parse status of any hypotheses leading to this response is also used, where $full\_parse \succ robust\_parse \succ no\_parse$.

Table 1 lists all of the features used to train the classifier, while Table 3 (in the appendix) lists the possible values for the non-numerical features. Figure 3 (in the appendix) gives an overview of the feature extraction process, as well as the classification method described in the next section.

6 Classifier Training and Scoring

For a given utterance, we now have a candidate list of responses derived from the speech recognizer's N-best list, a feature vector associated with each response, and a label telling us the "correct" response, as derived from the transcript. In order to build a classifier, we first label each response as either *acceptable* or *unacceptable* by comparing it to the system's response to the transcribed utterance. If the two responses are identical, then the response is labeled as *acceptable*; otherwise, it is labeled as *unacceptable*. This yields a binary decision problem for each response, given a set of features. We train a Support Vector Machine (SVM) to make this decision, using the Weka toolkit, version 3.4.12 (Witten and Frank, 2005).

Given a trained SVM model, the procedure for processing a novel utterance is as follows. First, classify each response (and its associated feature vector) on the response list for that utterance using the SVM. By using a logistic regression model fit on the training data, an SVM score between -1 and 1 for each response is yielded, where responses with positive scores are more likely to be *acceptable*, and those with negative scores are more likely to be *unacceptable*.

Next, the SVM scores are used to rank the list of responses. Given a ranked list of such responses, the dialogue system has two options: it can choose the top scoring response, or it can *abstain* from choosing any response. The most straightforward method for making such a decision is via a threshold: if the score of the top response is above a certain threshold, this response is accepted; otherwise, the system abstains from choosing a response, and instead responds by indicating non-understanding. Figure 3 (in the appendix) provides a graphical overview of the response confidence scoring process.

At first blush, a natural threshold to choose is 0, as this marks the boundary between *acceptable* and *unacceptable*. However, it may be desirable to optimize this threshold based on the desired characteristics of the dialogue system—in a mission-critical application, for example, it may be preferable to accept only high-confidence responses, and to clarify otherwise. We can optimize the threshold as we like using either the same training data, or a held-out development set, so long as we have an objective function with which to optimize. In the evaluation that follows, we optimize the threshold using the F-measure on the training data as the objective function. It would also be interesting to optimize the threshold in a more sophisticated manner, such as that developed in (Bohus and Rudnicky, 2005) where task success is used to derive the cost of misunderstandings and false rejections, which in turn are used to set a rejection threshold.

While a thresholding approach makes sense, other approaches are feasible as well. For instance, a second classifier could be used to decide whether or not to accept the top ranking response. The classifier could take into account such features as the spread in scores among the responses, the number classified as *acceptable*, the drop between the top score and the second-ranked score, *etc*.

7 Evaluation

We evaluated the response-based method using the data described in section 3, N-best lists with a maximum length of 10, and an SVM with a linear kernel. We note that, in the live system, two-pass recognition is performed for some utterances, in which a key concept recognized in the first pass (*e.g.*, a city name) causes a dynamic update to the contents of a class in the n-gram language model (*e.g.*, a set of street names) for the second pass—as in the utterance *Show me thirty two Vassar Street in Cambridge* where the city name (*Cambridge*) triggers a second pass in which the streets in that city are given a higher weight. This two-pass approach has been shown previously to decrease word and concept error rates (Gruenstein and Seneff, 2006), even though it can be susceptible to errors in understanding. However, since all street names, for example, are active in the vocabulary at all times, the two-pass approach is not strictly necessary to arrive at the correct hypotheses. Hence, for simplicity, in the experiments reported here, we do not integrate the two-pass approach—as this would require us to potentially do a second recognition pass for every candidate response. In a live system, a good strategy might be to consider a second recognition pass based on the top few candidate responses alone, which would produce a new set of candidates to be scored.

We performed 38-fold cross validation, where in each case the held-out test set was comprised of all the utterances of a single user. This ensured that we obtained an accurate prediction of a novel user's experience, although it meant that the test sets were not of equal size. We calculated F-measure for each test set, using the methodology described in figure 4 (in the appendix).

7.1 Baseline

As a baseline, we made use of the existing confidence module in the SUMMIT recognizer (Hazen et al., 2002). The module uses a linear projection model to produce an utterance level confidence score based on 15 features derived from recognizer scores,

Method	*F*
Recognition Confidence (Baseline)	.62
Recog Features Only	.62
Recog + Distributional	.67
Recog + Response	.71*
Recog + Response + Distributional	.72**

Table 2: Average F-measures obtained via per-user cross-validation of the response-based confidence scoring method using the feature sets described in Section 5, as compared to a baseline system which chooses the top hypothesis if the recognizer confidence score exceeds an optimized rejection threshold. The starred scores are a statistically significant (* indicates $p < .05$, ** indicates $p < .01$) improvement over the baseline, as determined by a paired t-test.

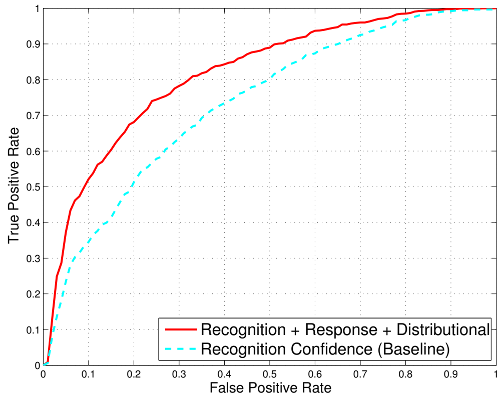

Figure 2: Receiver Operator Characteristic (ROC) curves (averaged across each cross-validation fold) comparing the baseline to the best response-based model.

and from comparing hypotheses on the N-best list. In our evaluation, the module was trained and tested on the same data as the SVM model using cross-validation.

An optimal rejection threshold was determined, as for the SVM method, using the training data with F-measure as the objective function. For each utterance, if the confidence score exceeded the threshold, then the response evoked from the top hypothesis on the N-best list was chosen.

7.2 Results

Table 2 compares the baseline recognizer confidence module to our response-based confidence annotator. The method was evaluated using several subsets of the features listed in Table 1. Using features derived from the recognizer only, we obtain results comparable to the baseline. Adding the response and distributional features yields a 16% improvement over the baseline system, which is statistically significant with $p < .01$ according to a paired t-test. While the distributional features appear to be helpful, the feature values derived from the response itself are the most beneficial, as they allow for a statistically significant improvement over the baseline when paired on their own with the recognizer-derived features.

Figure 2 plots ROC curves comparing the performance of the baseline model to the best response-based model. The curves were obtained by varying the value of the rejection threshold. We observe that the response-based model outperforms the baseline no matter what we set our tolerance for false positives to be.

The above results were obtained by using an SVM with a linear kernel, where feature values were normalized to be on the unit interval. We also tried using a quadratic kernel, retaining the raw feature values, and reducing the number of binary features by manually binning the non-numeric feature values. Each change resulted in a slight decrease in F-measure.

8 Conclusion and Future Work

We recast the problem of choosing among an N-best list of recognition hypotheses as one of choosing the best candidate system response which can be generated from the recognition hypotheses on that list. We then demonstrated a framework for assigning confidence scores to those responses, by using the scores output by an SVM trained to discriminate between acceptable and unacceptable responses. The classifier was trained using a set of features derived from the speech recognizer, culled from the generation of each response, and calculated based on each response's distribution. We tested our methods using data collected by users interacting with the City Browser multimodal dialogue system, and showed that they lead to a significant improvement over a baseline which makes an acceptance decision based on an utterance-level recognizer confidence score.

The technique developed herein could be refined in several ways. First and foremost, it may well be possible to find additional features with discriminatory power. Also, the decision as to whether or not to choose the top-scoring response could potentially be improved by choosing a more appropriate metric than F-measure as the objective function, or perhaps by using a second classifier at this stage.

Finally, our experiments were performed off-line. In order to better test the approach, we plan to deploy the classifier as a component in the running dialogue system. This presents some processing time constraints (as multiple candidate responses must be generated); and it introduces the confounding factor of working with a recognizer that can make multiple recognition passes after language model reconfiguration. These challenges should be tractable for N-best lists of modest length.

Acknowledgments

Thank you to Stephanie Seneff for her guidance and advice. Thanks to Timothy J. Hazen for his assistance with the confidence module. Thanks to Ali Mohammad for discussions about the machine learning aspects of this paper and his comments on drafts. And thanks to four anonymous reviewers for constructive criticism. This research is sponsored by the T-Party Project, a joint research program between MIT and Quanta Computer Inc., Taiwan.

References

Dan Bohus and Alex Rudnicky. 2002. Integrating multiple knowledge sources for utterance-level confidence annotation in the CMU Communicator spoken dialog system. Technical Report CS-190, Carnegie Mellon University.

Dan Bohus and Alexander I. Rudnicky. 2005. A principled approach for rejection threshold optimization in spoken dialog systems. In *Proc. of INTERSPEECH*.

Lin Chase. 1997. Word and acoustic confidence annotation for large vocabulary speech recognition. In *Proc. of 5th European Conference on Speech Communication and Technology*, pages 815–818.

Ananlada Chotimongkol and Alexander I. Rudnicky. 2001. N-best speech hypotheses reordering using linear regression. In *Proc. of 7th European Conference on Speech Communication and Technology*.

Grace Chung, Stephanie Seneff, Chao Wang, and Lee Hetherington. 2004. A dynamic vocabulary spoken dialogue interface. In *Proc. of INTERSPEECH*, pages 327–330.

Ed Filisko and Stephanie Seneff. 2003. A context resolution server for the Galaxy conversational systems. In *Proc. of EUROSPEECH*.

Malte Gabsdil and Oliver Lemon. 2004. Combining acoustic and pragmatic features to predict recognition performance in spoken dialogue systems. In *Proc. of Association for Computational Linguistics*.

Alexander Gruenstein and Stephanie Seneff. 2006. Context-sensitive language modeling for large sets of proper nouns in multimodal dialogue systems. In *Proc. of IEEE/ACL 2006 Workshop on Spoken Language Technology*.

Alexander Gruenstein and Stephanie Seneff. 2007. Releasing a multimodal dialogue system into the wild: User support mechanisms. In *Proc. of the 8th SIGdial Workshop on Discourse and Dialogue*, pages 111–119.

Alexander Gruenstein, Stephanie Seneff, and Chao Wang. 2006. Scalable and portable web-based multimodal dialogue interaction with geographical databases. In *Proc. of INTERSPEECH*.

Timothy J. Hazen, Stephanie Seneff, and Joseph Polifroni. 2002. Recognition confidence scoring and its use in speech understanding systems. *Computer Speech and Language*, 16:49–67.

Diane J. Litman, Julia Hirschberg, and Marc Swerts. 2000. Predicting automatic speech recognition performance using prosodic cues. In *Proc. of NAACL*, pages 218 – 225.

Michael Niemann, Sarah George, and Ingrid Zukerman. 2005. Towards a probabilistic, multi-layered spoken language interpretation system. In *Proc. of 4th IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, pages 8–15.

Rubén San-Segundo, Bryan Pellom, Wayne Ward, and José M. Pardo. 2000. Confidence measures for dialogue management in the CU Communicator System. In *Proc. of ICASSP*.

Stephanie Seneff. 1992. TINA: A natural language system for spoken language applications. *Computational Linguistics*, 18(1):61–86.

Stephanie Seneff. 2002. Response planning and generation in the MERCURY flight reservation system. *Computer Speech and Language*, 16:283–312.

Marilyn Walker, Jerry Wright, and Irene Langkilde. 2000. Using natural language processing and discourse features to identify understanding errors in a spoken dialogue system. In *Proc. 17th International Conf. on Machine Learning*, pages 1111–1118.

Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition.

Recognition N-best

Hypothesis	**Rank**	$\mathbf{S_t}$	$\mathbf{S_a}$	$\mathbf{S_l}$	**…**	***DS***	**Response**	**Parse**	**…**
thirty two vassal street in cambridge	0	45.3	28.5	26.5		→	R_0	*FULL*	
thirty two vassar street in cambridge	1	45.0	27.1	30.5		→	R_1	*FULL*	
thirty two vassar street in in cambridge	2	44.2	26.0	30.4		→	R_1	*ROBUST*	
at thirty two vassar street <noise>	3	40.1	26.5	29.4		→	R_1	*FULL*	
at thirty two vassal street in cambridge	4	39.5	26.3	29.0		→	R_1	*FULL*	
thirty two vassar street cambridge <noise>	5	38.4	25.8	28.4		→	R_1	*FULL*	
thirty two vassar street in canton	6	38.0	25.8	28.3		→	R_2	*FULL*	
thirty two vassal street in in canton	7	33.5	22.5	27.5		→	R_3	*ROBUST*	
twenty vassar in street in zoom	8	32.4	22.3	26.3		→	R_4	*NONE*	
thirty two vassar street in cambridge <noise>	9	32.0	19.5	26.7		→	R_1	*FULL*	

Response List

Response	**Rank**	$\mathbf{S_t}$	$\mathbf{S_a}$	$\mathbf{S_l}$	***%Top3***	***%Top5***	***Dist.***	***Parse***	**…**	***SVM***	**Score**	
R_0	0	45.3	28.5	26.5	.33	.8	5	*FULL*		→	.42	
R_1	1	45.0	27.1	30.5	.66	.2	5	*FULL*		→	.73	**→ R_1**
R_2	6	38.0	25.8	28.3	0.0	0.0	5	*FULL*		→	-.32	
R_3	7	33.5	22.5	27.5	0.0	0.0	5	*ROBUST*		→	-.55	
R_4	8	32.4	22.3	36.3	0.0	0.0	5	*NONE*		→	-.92	

Figure 3: The feature extraction and classification process. The top half of the digram shows how an N-best list of recognizer hypotheses, with associated scores from the recognizer, are processed by the dialogue system (*DS*) to produce a list of responses. Associated with each response is a set of feature values derived from the response itself, as well as the process of evoking the response (*e.g.* the parse status). The bottom half of the figure shows how the unique responses are collapsed into a list. Each response in the list inherits the best recognition scores available from hypotheses evoking that response; each also has feature values associated with it derived from the distribution of that response on the recognizer N-best list. Each set of feature values is classified by a Support Vector Machine, and the resulting score is used to rank the responses. If the highest scoring response exceeds the rejection threshold, then it is chosen as the system's response.

Feature	Possible Values
response_type top_response_type	geography, give_directions, goodbye, greetings, help_directions_did_not_understand_from_place, help_directions_did_not_understand_to_place, help_directions_no_to_or_from_place, help_directions_subway, hide_subway_map, history_cleared, list_cuisine, list_name, list_street, no_circled_data, no_data, no_match_near, non_unique_near, ok, panning_down, panning_east, panning_south, panning_up, panning_west, presupp_failure, provide_city_for_address, refined_result, reject_or_give_help, show_address, show_subway_map, speak_properties, speak_property, speak_verify_false, speak_verify_true, welcome_gui, zooming, zooming_in, zooming_out
POI_type	none, city, museum, neighborhood, restaurant, subway_station
parse_status	no_parse, robust_parse, full_parse
geographical_filter	none, address, circle, line, list_item, map_bounds, museum, neighborhood, point, polygon, restaurant, subway_station, city

Table 3: The set of possible values for non-numerical features, which are converted to sets of binary features.

Case I

R_0 is *acceptable* and is not *reject*

$S_0 \geq T \rightarrow$ T.P.
$S_0 < T \rightarrow$ F.N.

Response	Score	Type	Label
R_0	S_0	speak_property	acceptable
R_1	S_1	list_cuisine	unacceptable
R_2	S_2	speak_property	unacceptable

Case I: Example Ranked Response List

Case II

No candidate responses *acceptable*, or *acceptable* response is *reject*

(a)
R_0 is not *reject*
$S_0 \geq T \rightarrow$ F.P.
$S_0 < T \rightarrow$ T.N.

(b)
R_0 is *reject*
$S_0 \geq T \rightarrow$ T.N.
$S_0 < T \rightarrow$ T.N.

Response	Score	Type	Label
R_0	S_0	speak_property	unacceptable
R_1	S_1	list_cuisine	unacceptable
R_2	S_2	speak_property	unacceptable
R_3	S_3	reject	unacceptable
R_4	S_4	zooming_out	unacceptable

Case II: Example Ranked Response List

Case III

R_n (with $n > 0$) is *acceptable* and is not *reject*

(a)
R_0 is not *reject*
$S_0 \geq T \rightarrow$ F.P.
$S_0 < T \rightarrow$ F.N.

(b)
R_0 is *reject*
$S_0 \geq T \rightarrow$ F.N.
$S_0 < T \rightarrow$ F.N.

Response	Score	Type	Label
R_0	S_0	speak_property	unacceptable
R_1	S_1	list_cuisine	acceptable
R_2	S_2	speak_property	unacceptable
R_3	S_3	reject	unacceptable
R_4	S_4	zooming_out	unacceptable

Case III: Example Ranked Response List

Figure 4: Algorithm for calculating the F-measure confusion matrix of True Positives (T.P.), False Positives (F.P.), True Negatives (T.N.), and False Negatives (F.N.). The ranking technique described in this paper creates a list of candidate system responses ranked by their scores. The top scoring response is then *accepted* if its score exceeds a threshold T, otherwise all candidate responses are *rejected*. As such, the problem is not a standard binary decision. We show all possible outcomes from the ranking process, and note whether each case is counted as a T.P., F.P., T.N., or F.N. We note that given this algorithm for calculating the confusion matrix, no matter how we set the threshold T, F-measure will always be penalized if Case III occurs.