Machine Learning for Shallow Interpretation of User Utterances in Spoken Dialogue Systems

Piroska Lendvai
ILK Research Group
Tilburg University
The Netherlands
P.Lendvai@uvt.nl

Antal van den Bosch
ILK Research Group
Tilburg University
The Netherlands
Antal.vdnBosch@uvt.nl

Emiel Krahmer
Communication & Cognition
Tilburg University
The Netherlands
E.J.Krahmer@uvt.nl

Abstract

We investigate to what extent automatic learning techniques can be used for shallow interpretation of user utterances in spoken dialogue systems. This task involves dialogue act classification, shallow understanding and problem detection simultaneously. For this purpose we train both a rule-induction and a memory-based learning algorithm on a large set of surface features obtained by affordable means from an annotated corpus of human-machine dialogues. Using a pseudo-exhaustive search, the parameters of both algorithms are optimized. The shallow interpretation task turns out to be a difficult one, partly since there are 112 types of user answers. The best overall accuracy (exact match) obtained was 56.5%, which is a significant improvement over the baseline. The best average precision and recall for dialogue act classification was 82.8%, for classifying slot types 80.6% and for detecting communication problems 77.4%.

1 Introduction

In recent years there has been an increased interest in using statistical and machine learning approaches for the processing of user utterances in spoken dialogue systems. **Dialogue act classification** is an example for which this approach has been relatively successful. The purpose of this task is to determine what the underlying intention of a user utterance is (e.g., suggest, request, reject, etc.). Various techniques have been used for this purpose, including statistical language models (Reithinger and Maier, 1995), maximum entropy estimations (Choi et al., 1999), mixed stochastic techniques (Stolcke et al., 2000), Bayesian modelling (Keizer et al., 2002) and transformation-based learning (Samuel et al., 1998b).

Another task for which such approaches have been applied is automatic **problem detection.** Given that current speech recognizers may still make recognition errors, it is important to try and detect these problems as soon as possible. Various researchers, including (Hirschberg et al., 2001; Van den Bosch et al., 2001; Lendvai et al., 2002b), have shown that users signal problems when they become aware of them and that it is possible to detect communications problems with a high accuracy on the basis of such user signals.

Finally, for **processing** and **understanding** spoken user utterances, statistical techniques have also proven their usefulness, either in combination with rule-based grammars (e.g., Van Noord et al. (1999)) or without them (e.g., Nakano et al. (1999)).

Dialogue act classification, problem detection and understanding are all highly relevant for the processing of user utterances in spoken dialogue systems. Still, none of the approaches mentioned above address these tasks in combination. Such a combined approach would constitute a **shallow interpretation** module which provides clues for the dialogue manager about *semantic* aspects (such as

the contents of the user's utterance) and *pragmatic* aspects (the dialogue act and feedback about the status of the dialogue). If we would be able to correctly obtain such a representation, interaction with the dialogue system could improve, allowing the dialogue manager module to switch its strategy (e.g., to different error recovery or confirmation strategy) to adapt to the given situation. Arguably, generating this combined semantic-pragmatic interpretation is a difficult task since there are many ways in which these different clues can be combined. In addition, some of these clues will be hard to predict (e.g., whether a user will accept instead of correct a misrecognition, or which pieces of information a user will decide to provide or correct).

The goal of this paper is to investigate to what extent different machine learning approaches can be used for the purpose of such shallow interpretation. We use two learning techniques, namely **rule-induction** and **memory-based** learning. Both learners are trained on a large set of features derived from an annotated corpus of human-machine dialogues with a Dutch train timetable information system. The features come from different sources and are all low-level and directly available in most current spoken dialogue systems. In the experiments no explicit feature selection is performed. Our earlier results on problem detection (based on the same corpus) showed that the best results are obtained using all features (Lendvai et al., 2002b). In addition, the learning techniques themselves are capable to determine which features are beneficial for the learning task and which are not. We *do* perform an extensive search to estimate the optimal settings of both algorithms for the current task.

The remainder of this paper is organized as follows. In Section 2 we describe the corpus that was used and the labelling of the utterances. In Section 3 we describe the learning instances that were derived from the corpus and the general structure of the optimization and classification experiments. The results are given in Section 4. We conclude with some general remarks in Section 5.

2 Corpus and Labelling

The corpus used in our study consists of 3,738 pairs of system questions and user answers; in total 441 full dialogues (involving more than 400 different speakers). The dialogues were sampled from a range of telephone calls where users interacted with a Dutch train timetable information system. The dialogues are relatively short (2-10 turns). The system uses a mixed-initiative dialogue strategy that prompts the user to fill various slots. The system needs to have these slot values before it can perform a database query. At all times, the system gives immediate feedback to the user, via implicit or explicit verification, on what it has understood. Users of this system will therefore always become aware of eventual misunderstandings from the following system question.

The semantic structure of the system prompt and the shallow semantics of the user's response were hand-labelled in terms of a simple and straightforward tag set, derived from an earlier annotation for problem signalling performed by (Van den Bosch et al., 2001) and from structured semantic annotations of user anwers based on update expressions (Veldhuijzen van Zanten et al., 1999). The resulting tag set is unambiguous, thus system prompts and user answers can always be associated with exactly one representation. The number of different tags for system prompts is 106, that of user utterances is 112. The number of different occurring pairs of system prompts and user response is 664: there is no obvious mapping from systems prompts to user reactions.

System prompts are tagged in terms of dialogue acts and slots. Basic dialogue acts include asking a question (**Q**), explicit verification (**E**), repeating a prompt (**R**), asking a meta-question (**M**) and offering travel advice (final result, **Fr**). Implicit verification is represented as the simultaneous occurrence of a question and a verification (**Q;I**). The slots to be filled from the user input are departure and arrival station (**V** and **A** respectively), and the corresponding day, time of day (i.e., morning, noon or night) and hour (represented as **D**, **T** and **H** respectively). These time slots can be questioned together ("when", **Q_DTH**) or in isolation (e.g., "at what time" **Q_H**). In addition, the system can ask whether the user wants to have the travel advice repeated (repeat connection, **Q_Rc**), or whether the user would like to have information about another connection (**Q_Oc**), or an earlier or

later one, and so on.

The following are some tagged example system prompts (translated from Dutch); the general tag-format is **act_slot**, the parentical number indicates the frequency of a tag in our data.

(i) From where to where do you want to travel?
Q_VA (556)
(ii) When do you want to travel from Amsterdam to Tilburg?
Q_DTH;I_VA (358)
(iii) I am sorry but I didn't understand you. Could you repeat from where you want to travel to Schiphol?
RQ_V;RI_A (107)
(iv) I am sorry but I didn't understand you. Could you repeat your answer?
M (83)

User utterances are likewise represented as a combination of dialogue acts and slots. Users can give information ('slot-filling', **S**), provide an answer with explicitly uttering 'yes' (**Y**) or 'no'/'don't'/'not' (**N**), or accept incorrect information (**A**). The following are three different answers to the second system question above (numbers again indicate frequency of the tag). The general tag-format of the classes is **act_slot_situation**.

(i) Tomorrow.
S_D_ok (228)
(ii) No, not to Tilburg but to SCHIPHOL **!**
N;S_A_pr (16)
(iii) Today at eight in the evening.
A;S_DTH_pr (3)

The first answer is an input in an unproblematic situation (**ok**). The second one features an explicit 'no' and at the same time corrects the misrecognized slot. The third answer illustrates acceptance since the user does not correct the misrecognized slot (Tilburg) that is verified implicitly in a therefore problematic (**pr**) situation.

Our annotation scheme thus uses a separate marker associated with user utterances that *follow* a question–answer pair in which the answer caused some communication problem, for instance (and most often) because it was misrecognized. Therefore, the special marker identifies the point at which the user became aware of the communication problem, since he or she has just heared a system prompt not in accordance with the information just given in the previous answer.

In sum, the user tag represents jointly a high-level dialogue act (S, A, Y, N), a shallow semantic interpretation of the types of slots filled by the user, and a high-level pragmatic "awareness" flag of a communication problem. Appendix A illustrates a complete tagged dialogue of the corpus.

3 Learning Experiments

3.1 Feature representation

The shallow interpretation learning task can now be paraphrased as follows: given a user utterance in its preceding dialogue context, tag it with a semantic-pragmatic interpretation. This tag includes dialogue act information, information about the contents and whether or not a communication problem arose. In (Lendvai et al., 2002b) we studied the usefulness of a wide range of features for problem detection in spoken dialogue systems using machine learning. We utilize the same features for the current study as well. The features were extracted automatically both from the state of the system and from the recognized prosody and wording of the user's utterance and are listed in Table 1.

From the Dialogue Manager (DM) we use the words of the current and the previous prompt as well as the sequence of ten system prompt types. The latter can be seen as a (partial) representation of the dialogue history, showing (among other things) for which slots the system thinks it has acquired the correct value. Many studies on dialogue act classification also make use of the history of the computed classes of user input, e.g., Samuel et al. (1998a). However, we opted for not using this feature, in order to avoid cumulative error (Qu et al., 1997) originating from incorporating incorrect hypotheses. At the same time, using the correct tagged history of the user's utterances is not realistic since a spoken dialogue system can never have access to those on-line.

The features representing user utterances are derived from both the output of the Automatic Speech Recognition module (ASR) as well as the raw audio. The ASR output of this particular sys-

Aspect	Feature
DM: prompt	sequence of last 10 prompt types
DM: lexical	words in current and previous prompt
ASR: confidence	summed confidence score of most confident path in current word graph
ASR: branching	branching factor in the word graph of current and previous utterance
ASR: lexical	bag-of-words of previous and current user turn; most confident recognized string
Prosody: pitch	maximum and minimum F0; position of maximum and minimum; mean F0 and standard deviation
Prosody: energy	maximum energy (RMS); position of maximum; mean RMS and standard deviation
Prosody: duration	length of utterance in seconds; length of initial pause in frames
Prosody: tempo	number of syllables per second

Table 1: Overview of the employed features.

tem produced a word graph, containing various word hypotheses along with confidence scores indicating how sure the system is that it recognized a certain word correctly. From each word graph we stripped the recognized words (including the potentially incorrect ones) and encoded these as a 759 bits bag-of-words (BoW) vector. The 759 bits represent all words that occurred at least once in our corpus. In each BoW vector we indicate whether a word was present ('p') in the corresponding word graph or not ('a').

From the word graph we extracted the duration of the initial pause, the speech tempo, and the degree of branching. The initial pause in the utterance (the length of the silence that precedes the utterance) may cue the degree of hesitation of the user in responding, cf. (Krahmer et al., 2001). The speech tempo of the utterance corresponds to the number of uttered syllables per second.

The complexity (or *branching factor*) in the word graph was also calculated both for the current and the preceding utterance, characterizing the degree of confusion in the graph; much branching in the word graph can be an indication of system uncertainty or noisy user input. The confidence measurements of the ASR were also converted into a feature: we summed the confidence scores over the nodes of the overall most confident path for the user input.

Furthermore, we incorporate prosodic features in the learning since those have been reported to function well for problem detection purposes (see e.g., Hirschberg et al. (2001)). Non-standard prosody may be a signal of hyperarticulate speech which is typically associated with corrections (compare the way "Schiphol" is pronounced in the second user example utterance in Section 2). From the audio recordings of the corpus we automatically extracted loudness (in terms of RMS, i.e, root mean square energy), duration of the utterance from silence to silence and pitch (in terms of F0, i.e., fundamental frequency).

Each of the 3,738 user utterances in the corpus is represented as a vector consisting of 2,479 features. Each instance contains the tagged representation of the corresponding utterance; this is the tag to be predicted.

3.2 Learners

Two learning algorithms were used for the shallow interpretation task, a memory-based one and a rule-induction one. For the former we used the TiMBL software package, version 4.3 (Daelemans et al., 2002). TiMBL incorporates a variety of memory-based pattern classification algorithms, each with fine-tunable metrics. We chose for working with the IB1 algorithm only (the default in TiMBL), taking the classical k-nearest neighbor approach to classification. This k-NN algorithm looks for those instances among the training data that are most similar to the test instance, and extrapolates their majority outcome to the test

instance's class. Memory-based learning is often called "lazy" learning, because the classifier simply stores all training examples in memory, without abstracting away from individual instances in the learning process.

In contrast, our other classifier is a "greedy" learning algorithm, RIPPER (Cohen, 1995), version 1, release 2.4. This learner induces rule sets for each of the classes in the data, with built-in heuristics to maximize accuracy and coverage for each rule induced. This approach aims at discovering the regularities in the data, and represent it by the simplest possible rule set. Rules are by default induced first for low-frequency classes, leaving the most frequent class the default rule.

3.3 Experimental set-up

Training and testing was done by 10-fold cross-validation (CV), where re-sampling was carried out by means of dialogue-based partitioning, thereby ensuring that no material from the same dialogue could be part of both the training and the test set. The performance of the learners was evaluated according to four measures. (1) Predictive accuracy (the percentage of correctly tagged test instances). Note that this is based on *exact* matches. Thus if the learner hypothesizes A;S_DTH_ok and the correct tag is A;S_DTH_pr, this counts as an incorrect prediction. The other three measures are used to gain more insight in prediction of the components and involve the precision, recall, and F-score proportionally weighted over all tags of (2) the part of the tag representing the higher-level dialogue act (S, A, Y, or N), (3) the part of the tag representing the filled slots, and (4) the part of the tag representing the presence of a communication problem. The F-score represents the harmonic mean of precision and recall. We use the unweighted variant of the F-score, which is defined as $2PR/(P+R)$ (P = precision, R = recall) (van Rijsbergen, 1979). Before the actual experiments were performed, we performed a pseudo-exhaustive search for the optimal setting of both algorithms.

3.4 Pseudo-exhaustive parameter optimization

Both IB1 and RIPPER have parameters that bias their performance. Since it is unknown beforehand which parameter setting yields the best generalisation performance, and since it is not allowed to use test material to make that estimation, a reasonable remaining estimate can be made by performing experiments on the training material itself: e.g. to run a 10-fold experiment on each of the 90% training set splits within the overall 10-fold CV experiment (cf. Kohavi and John (1997)). One parameter setting can thus be tested by running 10 wrapped 10-fold CV experiments on training material, and averaging over the 100 test scores. This procedure can be repeated for other parameter settings, and the parameter setting with the highest estimated generalisation performance can then be selected to be applied to the full 90% training set, and tested on the yet unseen 10% test set.

The size of our data set and the available computer power enabled us to test a pseudo-exhaustive combination of parameter settings. The search is not truly exhaustive because we did not try the massive number of possible values for numeric parameters; we only tested the range of values we estimated to be reasonable. Nevertheless, our approach searches the space of possibilities considerably more thorough than economic search heuristics such as Monte Carlo sampling (Samuel et al., 1998a). With IB1 the following metrics were tested, amounting to 360 permuted combinations tested (for details, cf. Daelemans et al. (2002)):

- the number of nearest neighbours used for extrapolation were 1, 3, 5, 7, 9, 11, 13, 15, 19, and 25
- the distance weighting metric of the k nearest neighbors was either majority class voting, linearly-inversed distance weighting, inverse distance weighting, or exponential-decay distance weighting with α set to 1, 2, or 4
- for computing the similarity between features either the overlap function or the modified value difference metric (MVDM) function was used
- for estimating the importance of the attributes in the classification task either no weighting,

	accuracy	dialogue act			filled slot types			comm. problems		
	(%)	pre	rec	F	pre	rec	F	pre	rec	F
baseline	13.3	23.1	20.8	21.9	—	0.0	—	—	0.0	—
	1.9	2.0	1.8	1.9	—	0.0	—	—	0.0	—
one-feature	47.7	82.8	75.7	79.4	87.8	71.8	78.9	73.3	52.2	61.0
	3.4	2.8	2.1	2.6	1.9	3.4	2.3	5.0	5.6	5.4

Table 2: Baseline and simple learning (i.e., one feature only) scores on shallow semantic interpretation, averaged over 10-fold CV experiments: accuracy, and proportionally weighted precision, recall and F-score on dialogue act type, filled slot types, and communication problems. Each second line shows standard deviation.

Information Gain, Gain Ratio, Chi-squared, or shared-variance weighting was used.

For the RIPPER algorithm the learners to be optimized were created by systematically varying the following parameters and their values, totalling 24 permuted combinations tested per parameter setting:

- negative tests on the feature attributes were either allowed or disallowed
- the number of optimization rounds on the induced ruleset was 0, 1, 2, 3
- the amount of learning instances to be minimally covered by each rule was set to 1, 2, 5, and 10
- the coding cost of a hypothesis was allowed to be multiplied by 0.5, 1.0, and 2.0.

3.5 Baselines

The straightforward baseline is to always predict the majority class. The most frequent tag among the 3738 user utterances in the corpus is N_ok (the user utters a negative lexical item but it does not signal a problem: "no, thank you"). This occurs 499 times. The strategy of always predicting this label yields 13.3% accuracy (**baseline**). In a way, this majority class baseline is misleading; a negative answer is much more likely following a yes/no question-type prompt such as Q_Oc ("Do you want to know another connection?") than it is following a question which involves various slots (e.g., arrival and destination, Q_VA). An alternative, directly learnable from the data, is to predict user input classes solely on the basis of one single feature: the most recently asked system prompt.

Always guessing the tag occuring most frequently in response to the last system prompt type (based on the 90% training sets, in the same 10-fold partitions as used by the learners) produces a baseline of 47.7% accuracy, an F-score of 79.4% on predicting the higher-level tags (S, A, Y, and N), an F-score of 78.9% on predicting the types of filled slots, and an F-score of 61.0% on the detection of communication problems. This learning experiment provides us with a very sharp baseline, but is in a way more informative than the majority class baseline. Details (including precision and recall) are given in Table 2. Generally, precision scores are higher than those of recall.

4 Results

Table 3 displays the performance of the two learners on the shallow interpretation task. As was to be expected, both learning methods perform significantly better than the majority class baseline. However, if we compare the results with those obtained by training on only a single feature (the most recent system prompt type), a more interesting picture emerges. On accuracy (i.e., the strongest, exact match criterion), both learners outperform the one-feature learner; IB1 reduces error by 17%, RIPPER by 11%. Both these accuracy scores are statistically significant (IB1: $t = 10.50, p < 0.01$, RIPPER: $t = 4.61, p < 0.01$).

If we look at the more detailed sub-measures, we see that only IB1 performs somewhat better than the one feature learner for dialogue act classification (an F-score of 82.8% vs. 79.4%) and on predicting the type of filled slots (an F-score of 80.6% vs. 78.9%). Interestingly, for both tasks

algorithm	accuracy (%)	dialogue act pre	rec	F	filled slot types pre	rec	F	comm. problems pre	rec	F
IB1	56.5	86.3	79.5	82.8	86.4	75.7	80.6	88.0	69.3	77.4
	2.7	2.5	2.2	2.2	2.1	2.9	2.0	3.6	4.1	3.0
RIPPER	53.6	79.3	75.3	77.2	84.5	70.5	76.9	83.0	65.0	72.7
	3.3	3.2	2.8	2.9	2.8	3.8	3.1	3.5	7.1	5.3

Table 3: Scores produced by IB1 and RIPPER on shallow semantic interpretation, averaged over 10-fold CV experiments: accuracy, and proportionally weighted precision, recall and F-score on dialogue act type, filled slot types, and communication problems. Each second line shows standard deviation.

RIPPER (trained on all features) performs under the one-feature learner. Only on problem detection, both IB1 and RIPPER improve over the baseline by a broad margin: against the baseline F-score of 61.0% IB1 attains 77.4%, and RIPPER attains 72.7%.

The one-feature learner performs so well because there appear to be strong correlations between system prompts and typical user answers that follow it (Lendvai et al., 2002a). This is not surprising; the hard part of the task seems to predict those cases where the user gives a different response than what is most likely. This is where IB1 and RIPPER can in principle do better than the baseline, using information provided by the context features. For the dialogue act type and the filled slots type, only IB1 is able to improve over the baseline, suggesting that this algorithm is able to use information in all other features besides the most recent system prompt to positive effect, whereas RIPPER is unable to utilize such information. The reason why the prediction of communication problems at the "aware" point is done better by the two learners than the baseline is that it is fairly unpredictable when a communication problem occurs; at least, it is less strongly related to the most recent type of system prompt. However, as (Van den Bosch et al., 2001) and (Lendvai et al., 2002b) report, IB1 and RIPPER can attain higher scores on the same corpus —F-scores between 87% and 90%— on detecting communication problems at the "aware" point when trained on that particular problem in isolation.

The near-exhaustive parameter optimization lead to the following settings for IB1: $k = 13$, the MVDM distance function, gain-ratio feature weighting, and linear-inversed distance weighting.

For RIPPER the optimal estimation was to allow negation, optimize two rounds, set the coding cost factor to 1.0, use default rule ordering, and cover a minimal number of 1 instance per rule. The latter setting allows for instance-specific rules, which is effectively close to IB1's instance-specific k-NN classification. Apparently, having instance-specific rules is optimal for this data, suggesting that there are one-instance exceptions that reoccur in test material. Having such exceptions as rules may pay off, and when they do not reoccur, such rules are generally harmless. In general, we see that the algorithm makes use of all kinds of features provided to it. It is noteworthy that out of the 270 rules generated by RIPPER on the total corpus material only about 30% cover five or more instances. In Appendix B a selection from the generated set of rules are given and explained.

5 Concluding remarks

We investigated the learnability of shallow semantic interpretation of user utterances in a train information dialogue system by two machine learning algorithms. We find that guessing the most frequent user semantics given a system prompt is a strategy that is hard to beat, but still leaves a considerable margin of error. Only the memory-based learner IB1 improved over this simple strategy in accuracy and performance on the three subtasks. The rule induction algorithm RIPPER performed worse on dialogue acts and slots than the one-feature strategy, apparently hindered by the fact that it was unable to learn rules for many user answer types, due to the lack of sufficient examples of these types, and reliable regularities in them.

The positive results with IB1 do suggest that shallow semantic interpretation from surface context features (system prompts, word graphs, user prosody) in annotated data is possible, also when dialogue act classification, slot filling, and problem detection are treated together as one shallow interpretation task. Our results leave to be investigated how well the learners would perform on each of the tasks in isolation. For problem detection, we already know performance is improved when the task is isolated. For the dialogue act classification and slot filling task, earlier work on the same train information data can serve as comparative material (Veldhuijzen van Zanten et al., 1999) if the slot-filling task is extended to identifiying the slot values.

The method described in this paper is generally applicable as it is language and task independent. Given enough data, it is likely that the method is more efficient in construction and running time than its counterpart that uses full parsing and/or full semantic analysis. We hypothesize that with the use of additional features the current performance would improve, so that eventually it could be integrated and tested in the context of a spoken dialogue system to achieve more refined strategies for interpreting user input.

References

W. Choi, J. Cho, and J. Sea. 1999. Analysis system of speech acts and discourse structures using maximum entropy model. In *Proc. of the 37th Annual Meeting of the Association for Computational Linguistics*.

W. W. Cohen. 1995. Fast effective rule induction. In *Proc. of the Twelfth International Conference on Machine Learning*, Lake Tahoe, California.

W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch. 2002. TiMBL: Tilburg memory based learner, version 4.3, reference guide. ILK technical report, Tilburg University. available from http://ilk.uvt.nl.

J. Hirschberg, D. Litman, and M. Swerts. 2001. Identifying user corrections automatically in spoken dialogue systems. In *Proc. of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL 2001)*.

S. Keizer, R. op den Akker, and A. Nijholt. 2002. Dialogue act recognition with Bayesian networks for Dutch dialogues. In *Proc. of 3rd SIGdial Workshop on Discourse and Dialogue*.

R. Kohavi and G. John. 1997. Wrappers for Feature Subset Selection. In *Artificial Intelligence:97(1-2):273-324*.

E. Krahmer, M. Swerts, M. Theune, and M. Weegels. 2001. The dual of denial: Two uses of disconfirmations in dialogue and their prosodic correlates. *Speech Communication*, 36(1):133–145.

P. Lendvai, A. Van den Bosch, E. Krahmer, and M. Swerts. 2002a. Improving machine-learned detection of miscommunications in human-machine dialogues through informed data splitting. In *Proc. ESSLLI'02 Workshop on Machine Learning Approaches in Computational Linguistics*.

P. Lendvai, A. Van den Bosch, E. Krahmer, and M. Swerts. 2002b. Multi-feature error detection in spoken dialogue systems. In *Proc. Computational Linguistics in the Netherlands (CLIN '01)*. Rodopi Amsterdam.

M. Nakano, N. Miyazaki, J. Hirasawa, K. Dohsaka, and T. Kawabata. 1999. Understanding unsegmented user utterances in real-time spoken dialogue systems. In *Proc. of the 37th Annual Meeting of the Association for Computational Linguistics*.

Y. Qu, B. DiEugenio, A. Lavie, L. Levin, and C. Rose, 1997. *Dialogue Processing in Spoken Language Systems: Revised Papers from ECAI-96 Workshop*, chapter "Minimizing Cumulative Error in Discourse Context". Springer Verlag.

N. Reithinger and E. Maier. 1995. Utilizing statistical dialogue act processing in verbmobil. In *Proceedings of the ACL*.

K. Samuel, S. Carberry, and K. Vijay-Shanker. 1998a. Computing dialogue acts from features with transformation-based learning. In *Proc. of the AAAI Spring Symposium on Applying Machine Learning to Discourse Processing*, pages 90–97.

K. Samuel, S. Carberry, and K. Vijay-Shanker. 1998b. Dialogue act tagging with transformation-based learning. In *Proc. of COLING/ACL*, pages 1150–1156.

A. Stolcke, K. Ries, N. Coccaro, E. Shriberg, R. Bates, D. Jurafsky, P. Taylor, R. Martin, C. Van Ess-Dykema, and M. Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26:339–373.

A. Van den Bosch, E. Krahmer, and M. Swerts. 2001. Detecting problematic turns in human-machine interactions: Rule-induction versus memory-based learning approaches. In *Proc. of the 39th Meeting of the Association for Computational Linguistics*, pages 499–506, New Brunswick, NJ. ACL.

G. van Noord, G. Bouma, R. Koeling, and M. Nederhof. 1999. Robust grammatical analysis for spoken dialogue systems. *Journal of Natural Language Engineering*, 5(1):45–93.

C.J. van Rijsbergen. 1979. *Information Retrieval*. Buttersworth, London.

G. Veldhuijzen van Zanten, G. Bouma, K. Sima'an, G. van Noord, and R. Bonnema. 1999. Evaluation of the NLP components of the OVIS2 spoken dialogue system. In *Computational Linguistics in the Netherlands 1998*, pages 213–229. Rodopi Amsterdam.

Appendix A: An example annotated dialogue - English translation

Turn #	Turn taker	Utterance	Semantic tag
1	System	good morning. this is the automatic information system of public transportation. this system provides information exclusively about train travels from NS-station to NS-station. from which station to which station would you like to travel?	Q_VA
	User	I want to go from Voorschoten to the Hague central station	S_VA_ok
2	System	when do you want to travel from Voorschoten to Utrecht central station?	Q_DTH;I_VA
	User	I want to go to the Hague central station	S_A_pr
3	System	when do you want to travel from Voorschoten to the Hague central station?	Q_DTH;I_VA
	User	the day after tomorrow	S_D_ok
4	System	at what time do you want to travel the day after tomorrow?	Q_H;I_D
	User	somewhere in the evening	S_T_ok
5	System	so you want to travel between eightteen and twenty four hours?	E_T
	User	yes	Y_ok
6	System	I am sorry I did not understand you. could you repeat whether you want to depart between eightteen and twenty four hours?	RE_T
	User	yes	Y_pr
7	System	there are five connections. this is the earliest. by passanger train to (...). do you want me to repeat the connection?	Fr;Q_Rc
	User	yes repeat it	Y_ok
8	System	I am sorry I did not understand you. could you say again if I need to repeat the connection?	RQ_Rc
	User	yes	Y_pr
9	System	I am sorry but again I did not understand you. could you say again if I need to repeat the connection?	RQ_Rc
	User	yes	Y_pr
10	System	I repeat the connection. by passanger train to (...). do you want me to repeat the connection?	Rc;Q_Rc
	User	no	N_ok
11	System	I am sorry I did not understand you. could you say again if I need to repeat the connection?	RQ_Rc
	User	no don't repeat	N_pr
12	System	I am sorry but again I did not understand you. could you say again if I need to repeat the connection?	RQ_Rc
	User	no	N_pr
13	System	do you want to know the next connection?	Q_Nxc
	User	no	N_ok
14	System	do you want to know another connection?	Q_Oc
	User	no	N_ok
15	System	I am sorry I did not understand you. could you say again if you want to know another connection?	RQ_Oc
	User	no I don't want to know another connection	N_pr

Appendix B: Selected RIPPER rules

The structure of a rule is: **if** <feature test> (**AND** <feature test>, etc.) **then** *class* (n/m), where <feature test> is a test on the presence of a nominal feature value, the presence of an element of a set feature, or a range of a numeric feature. n indicates the number of instances a rule covers, m the number of false predictions.
Lexical items are translated from Dutch into English. Feature codes are: *promptC*: current prompt; *sysC*: current system utterance; *usrC/P*: current/previous user utterance; *rmsstdev, rmsmean, ipausedur* are prosodic features (see Table 1).
Rule 1 reveals that very often users are not correcting repeatedly misunderstood system prompts, but instead accept (*A*) the erroneous implicit verification (the RI_D part of *promptC* in a *pr* environment) and provide information for the prompted slots (*S_VA* in reply to RQ_VA), hoping to repair the misunderstanding at a later point.
Rule 2 captures the characteristics of another type of acceptance, when the user answers with explicit 'yes' to the system's confirmation of the misunderstood time slot. Interestingly, the learner uses a prosodic feature (the standard deviation in the loudness of the input) to base its prediction on.
In Rule 3 we see that if the more emphatic 'where to' interrogative is used by the system, the user answer will often contain arrival information only, and this happens in problematic situations.
Rule 4 is fairly general, hypothesizing that whenever the system asks for the time of travel, the user will provide it without experiencing misunderstandings (*ok*), and indeed this rule has very low precision.
Hypothesis 5 describes a dialogue situation when the system talks about itself ('I' is present), prompting for time information ('o'clock' is present), whereas in the previous turn the user has provided that ('o'clock' is in the word graph). Nonetheless, the user is providing the information (*S_H*) again.
Rule 6 sheds light on the unproblematic circumstances of giving the departure slot value: the user did not yet say this in the previous turn ('to' was absent in that word graph).
The same slot is filled with a certain amount of hesitation (signalled by the the duration of the initial pause feature) under problematic circumstances in Rule 7.
A good strategy from the classifier is to assume an unproblematic 'yes' answer when certain words ('to', 'no', 'from') were not recognized in the input but 'yes' was present in the word graph (Rule 8).

1. **if** promptC = RQ_VA;RI_D **then** *A;S_VA_pr* (7/1)
2. **if** 'so' ∈ sysC ∧ promptC = E_T ∧ 'o'clock' ∈ usrP ∧ rmsstdev >= 874 **then** *A;Y_pr* (10/0)
3. **if** 'where to' ∈ sysC **then** *S_A_pr* (35/7)
4. **if** 'time' ∈ sysC **then** *S_H_ok* (97/75)
5. **if** 'time' ∈ sysC ∧ 'I' ∈ sysC ∧ 'o'clock' ∈ usrP **then** *S_H_pr* (25/7)
6. **if** promptC = Q_V;E_A ∧ 'to' ∉ usrP ∧ rmsmean <= 240 **then** *S_V_ok* (10/0)
7. **if** 'from' ∈ sysC ∧ 'to' ∉ usrC ∧ ipausedur >= 12 **then** *S_V_pr* (68/5)
8. **if** 'to' ∉ sysC ∧ 'no' ∉ usrC ∧ 'from' ∉ sysC ∧ 'yes' ∈ usrC **then** *Y_ok* (82/12)