Evaluating language understanding accuracy with respect to objective outcomes in a dialogue system

Myroslava O. Dzikovska and **Peter Bell** and **Amy Isard** and **Johanna D. Moore**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh, United Kingdom
{m.dzikovska,peter.bell,amy.isard,j.moore}@ed.ac.uk

Abstract

It is not always clear how the differences in intrinsic evaluation metrics for a parser or classifier will affect the performance of the system that uses it. We investigate the relationship between the intrinsic evaluation scores of an interpretation component in a tutorial dialogue system and the learning outcomes in an experiment with human users. Following the PARADISE methodology, we use multiple linear regression to build predictive models of learning gain, an important objective outcome metric in tutorial dialogue. We show that standard intrinsic metrics such as F-score alone do not predict the outcomes well. However, we can build predictive performance functions that account for up to 50% of the variance in learning gain by combining features based on standard evaluation scores and on the confusion matrix entries. We argue that building such predictive models can help us better evaluate performance of NLP components that cannot be distinguished based on F-score alone, and illustrate our approach by comparing the current interpretation component in the system to a new classifier trained on the evaluation data.

1 Introduction

Much of the work in natural language processing relies on intrinsic evaluation: computing standard evaluation metrics such as precision, recall and F-score on the same data set to compare the performance of different approaches to the same NLP problem. However, once a component, such as a parser, is included in a larger system, it is not always clear that improvements in intrinsic evaluation scores will translate into improved overall system performance. Therefore, extrinsic or task-based evaluation can be used to complement intrinsic evaluations. For example, NLP components such as parsers and co-reference resolution algorithms could be compared in terms of how much they contribute to the performance of a textual entailment (RTE) system (Sammons et al., 2010; Yuret et al., 2010); parser performance could be evaluated by how well it contributes to an information retrieval task (Miyao et al., 2008).

However, task-based evaluation can be difficult and expensive for interactive applications. Specifically, task-based evaluation for dialogue systems typically involves collecting data from a number of people interacting with the system, which is time-consuming and labor-intensive. Thus, it is desirable to develop an off-line evaluation procedure that relates intrinsic evaluation metrics to predicted interaction outcomes, reducing the need to conduct experiments with human participants.

This problem can be addressed via the use of the PARADISE evaluation methodology for spoken dialogue systems (Walker et al., 2000). In a PARADISE study, after an initial data collection with users, a performance function is created to predict an outcome metric (e.g., user satisfaction) which can normally only be measured through user surveys. Typically, a multiple linear regression is used to fit a predictive model of the desired metric based on the values of interaction parameters that can be derived from system logs without additional user studies (e.g., dialogue length, word error rate, number of misunderstandings).

PARADISE models have been used extensively in task-oriented spoken dialogue systems to establish which components of the system most need improvement, with user satisfaction as the outcome metric (Möller et al., 2007; Möller et al., 2008; Walker et al., 2000; Larsen, 2003). In tutorial dialogue, PARADISE studies investigated

which manually annotated features predict learning outcomes, to justify new features needed in the system (Forbes-Riley et al., 2007; Rotaru and Litman, 2006; Forbes-Riley and Litman, 2006).

We adapt the PARADISE methodology to evaluating individual NLP components, linking commonly used intrinsic evaluation scores with extrinsic outcome metrics. We describe an evaluation of an interpretation component of a tutorial dialogue system, with student learning gain as the target outcome measure. We first describe the evaluation setup, which uses standard classification accuracy metrics for system evaluation (Section 2). We discuss the results of the intrinsic system evaluation in Section 3. We then show that standard evaluation metrics do not serve as good predictors of system performance for the system we evaluated. However, adding confusion matrix features improves the predictive model (Section 4). We argue that in practical applications such predictive metrics should be used alongside standard metrics for component evaluations, to better predict how different components will perform in the context of a specific task. We demonstrate how this technique can help differentiate the output quality between a majority class baseline, the system's output, and the output of a new classifier we trained on our data (Section 5). Finally, we discuss some limitations and possible extensions to this approach (Section 6).

2 Evaluation Procedure

2.1 Data Collection

We collected transcripts of students interacting with BEETLE II (Dzikovska et al., 2010b), a tutorial dialogue system for teaching conceptual knowledge in the basic electricity and electronics domain. The system is a learning environment with a self-contained curriculum targeted at students with no knowledge of high school physics. When interacting with the system, students spend 3-5 hours going through pre-prepared reading material, building and observing circuits in a simulator, and talking with a dialogue-based computer tutor via a text-based chat interface.

During the interaction, students can be asked two types of questions. Factual questions require them to name a set of objects or a simple property, e.g., "Which components in circuit 1 are in a closed path?" or "Are bulbs A and B wired in series or in parallel". Explanation and definition questions require longer answers that consist of 1-2 sentences, e.g., "Why was bulb A on when switch Z was open?" (expected answer "Because it was still in a closed path with the battery") or "What is voltage?" (expected answer "Voltage is the difference in states between two terminals"). We focus on the performance of the system on these long-answer questions, since reacting to them appropriately requires processing more complex input than factual questions.

We collected a corpus of 35 dialogues from paid undergraduate volunteers interacting with the system as part of a formative system evaluation. Each student completed a multiple-choice test assessing their knowledge of the material before and after the session. In addition, system logs contained information about how each student's utterance was interpreted. The resulting data set contains 3426 student answers grouped into 35 subsets, paired with test results. The answers were then manually annotated to create a gold standard evaluation corpus.

2.2 BEETLE II Interpretation Output

The interpretation component of BEETLE II uses a syntactic parser and a set of hand-authored rules to extract the domain-specific semantic representations of student utterances from the text. The student answer is first classified with respect to its domain-specific speech act, as follows:

- Answer: a contentful expression to which the system responds with a tutoring action, either accepting it as correct or remediating the problems as discussed in (Dzikovska et al., 2010a).
- Help request: any expression indicating that the student does not know the answer and without domain content.
- Social: any expression such as "sorry" which appears to relate to social interaction and has no recognizable domain content.
- Uninterpretable: the system could not arrive at any interpretation of the utterance. It will respond by identifying the likely source of error, if possible (e.g., a word it does not understand) and asking the student to rephrase their utterance (Dzikovska et al., 2009).

If the student utterance was determined to be an answer, it is further diagnosed for correctness as discussed in (Dzikovska et al., 2010b), using a domain reasoner together with semantic representations of expected correct answers supplied by human tutors. The resulting diagnosis contains the following information:

- Consistency: whether the student statement correctly describes the facts mentioned in the question and the simulation environment: e.g., student saying "Switch X is closed" is labeled inconsistent if the question stipulated that this switch is open.
- Diagnosis: an analysis of how well the student's explanation matches the expected answer. It consists of 4 parts
  - Matched: parts of the student utterance that matched the expected answer
  - Contradictory: parts of the student utterance that contradict the expected answer
  - Extra: parts of the student utterance that do not appear in the expected answer
  - Not-mentioned: parts of the expected answer missing from the student utterance.

The speech act and the diagnosis are passed to the tutorial planner which makes decisions about feedback. They constitute the output of the interpretation component, and its quality is likely to affect the learning outcomes, therefore we need an effective way to evaluate it. In future work, performance of individual pipeline components could also be evaluated in a similar fashion.

2.3 Data Annotation

The general idea of breaking down the student answer into correct, incorrect and missing parts is common in tutorial dialogue systems (Nielsen et al., 2008; Dzikovska et al., 2010b; Jordan et al., 2006). However, representation details are highly system specific, and difficult and time-consuming to annotate. Therefore we implemented a simplified annotation scheme which classifies whole answers as correct, partially correct but incomplete, or contradictory, without explicitly identifying the correct and incorrect parts. This makes it easier to create the gold standard and still retains useful information, because tutoring systems often choose the tutoring strategy based on the general answer class (correct, incomplete, or contradictory). In addition, this allows us to cast the problem in terms of classifier evaluation, and to use standard classifier evaluation metrics. If more detailed annotations were available, this approach could easily be extended, as discussed in Section 6.

We employed a hierarchical annotation scheme shown in Figure 1, which is a simplification of the DeMAND coding scheme (Campbell et al., 2009). Student utterances were first annotated as either related to domain content, or not containing any domain content, but expressing the student's metacognitive state or attitudes. Utterances expressing domain content were then coded with respect to their correctness, as being fully correct, partially correct but incomplete, containing some errors (rather than just omissions) or irrelevant[1]. The "irrelevant" category was used for utterances which were correct in general but which did not directly answer the question. Inter-annotator agreement for this annotation scheme on the corpus was $\kappa = 0.69$.

The speech acts and diagnoses logged by the system can be automatically mapped into our annotation labels. Help requests and social acts are assigned the "non-content" label; answers are assigned a label based on which diagnosis fields were filled: "contradictory" for those answers labeled as either inconsistent, or containing something in the contradictory field; "incomplete" if there is something not mentioned, but something matched as well, and "irrelevant" if nothing matched (i.e., the entire expected answer is in not-mentioned). Finally, uninterpretable utterances are treated as unclassified, analogous to a situation where a statistical classifier does not output a label for an input because the classification probability is below its confidence threshold.

This mapping was then compared against the manually annotated labels to compute the intrinsic evaluation scores for the BEETLE II interpreter described in Section 3.

3 Intrinsic Evaluation Results

The interpretation component of BEETLE II was developed based on the transcripts of 8 sessions

[1]Several different subcategories of non-content utterances, and of contradictory utterances, were recorded. However, they resulting classes were too small and so were collapsed into a single category for purposes of this study.

Category	Subcategory	Description
Non-content		Metacognitive and social expressions without domain content, e.g., "I don't know", "I need help", "you are stupid"
Content		The utterance includes domain content.
	correct	The student answer is fully correct
	pc_incomplete	The student said something correct, but incomplete, with some parts of the expected answer missing
	contradictory	The student's answer contains something incorrect or contradicting the expected answer, rather than just an omission
	irrelevant	The student's statement is correct in general, but it does not answer the question.

Figure 1: Annotation scheme used in creating the gold standard

Label	Count	Frequency
correct	1438	0.43
pc_incomplete	796	0.24
contradictory	808	0.24
irrelevant	105	0.03
non_content	232	0.07

Table 1: Distribution of annotated labels in the evaluation corpus

of students interacting with earlier versions of the system. These sessions were completed prior to the beginning of the experiment during which our evaluation corpus was collected, and are not included in the corpus. Thus, the corpus constitutes unseen testing data for the BEETLE II interpreter.

Table 1 shows the distribution of codes in the annotated data. The distribution is unbalanced, and therefore in our evaluation results we use two different ways to average over per-class evaluation scores. Macro-average combines per-class scores disregarding the class sizes; micro-average weighs the per-class scores by class size. The overall classification accuracy (defined as the number of correctly classified instances out of all instances) is mathematically equivalent to micro-averaged recall; however, macro-averaging better reflects performance on small classes, and is commonly used for unbalanced classification problems (see, e.g., (Lewis, 1991)).

The detailed evaluation results are presented in Table 2. We will focus on two metrics: the overall classification accuracy (listed as "micro-averaged recall" as discussed above), and the macro-averaged F score.

The majority class baseline is to assign "correct" to every instance. Its overall accuracy is 43%, the same as BEETLE II. However, this is obviously not a good choice for a tutoring system, since students who make mistakes will never get tutoring feedback. This is reflected in a much lower value of the F score (0.12 macroaverage F score for baseline vs. 0.44 for BEETLE II). Note also that there is a large difference in the micro- and macro- averaged scores. It is not immediately clear which of these metrics is the most important, and how they relate to actual system performance. We discuss machine learning models to help answer this question in the next section.

4 Linking Evaluation Measures to Outcome Measures

Although the intrinsic evaluation shows that the BEETLE II interpreter performs better than the baseline on the F score, ultimately system developers are not interested in improving interpretation for its own sake: they want to know whether the time spent on improvements, and the complications in system design which may accompany them, are worth the effort. Specifically, do such changes translate into improvement in overall system performance?

To answer this question without running expensive user studies we can build a model which predicts likely outcomes based on the data observed so far, and then use the model's predictions as an additional evaluation metric. We chose a multiple linear regression model for this task, linking the classification scores with learning gain as measured during the data collection. This approach follows the general PARADISE approach (Walker et al., 2000), but while PARADISE is typically used to determine which system components need

Label	baseline			BEETLE II		
	prec.	recall	F1	prec.	recall	F1
correct	0.43	1.00	0.60	0.93	0.52	0.67
pc_incomplete	0.00	0.00	0.00	0.42	0.53	0.47
contradictory	0.00	0.00	0.00	0.57	0.22	0.31
irrelevant	0.00	0.00	0.00	0.17	0.15	0.16
non-content	0.00	0.00	0.00	0.91	0.41	0.57
macroaverage	0.09	0.20	0.12	0.60	0.37	0.44
microaverage	0.18	0.43	0.25	0.70	0.43	0.51

Table 2: Intrinsic Evaluation Results for the BEETLE II and a majority class baseline

the most improvement, we focus on finding a better performance metric for a single component (interpretation), using standard evaluation scores as features.

Recall from Section 2.1 that each participant in our data collection was given a pre-test and a post-test, measuring their knowledge of course material. The test score was equal to the proportion of correctly answered questions. The normalized learning gain, $\frac{post-pre}{1-pre}$ is a metric typically used to assess system quality in intelligent tutoring, and this is the metric we are trying to model.

Thus, the training data for our model consists of 35 instances, each corresponding to a single dialogue and the learning gain associated with it. We can compute intrinsic evaluation scores for each dialogue, in order to build a model that predicts that student's learning gain based on these scores. If the model's predictions are sufficiently reliable, we can also use them for predicting the learning gain that a student could achieve when interacting with a new version of the interpretation component for the system, not yet tested with users. We can then use the predicted score to compare different implementations and choose the one with the highest predicted learning gain.

4.1 Features

Table 4 lists the feature sets we used. We tried two basic types of features. First, we used the evaluation scores reported in the previous section as features. Second, we hypothesized that some errors that the system makes are likely to be worse than others from a tutoring perspective. For example, if the student gives a contradictory answer, accepting it as correct may lead to student misconceptions; on the other hand, calling an irrelevant answer "partially correct but incomplete" may be less of a problem. Therefore, we computed separate confusion matrices for each student. We normalized each confusion matrix cell by the total number of incorrect classifications for that student. We then added features based on confusion frequencies to our feature set.[2]

Ideally, we should add 20 different features to our model, corresponding to every possible confusion. However, we are facing a sparse data problem, illustrated by the overall confusion matrix for the corpus in Table 3. For example, we only observed 25 instances where a contradictory utterance was miscategorized as correct (compared to 200 "contradictory–pc_incomplete" confusions), and so for many students this misclassification was never observed, and predictions based on this feature are not likely to be reliable. Therefore, we limited our features to those misclassifications that occurred at least twice for each student (i.e., at least 70 times in the entire corpus). The list of resulting features is shown in the "conf" row of Table 4. Since only a small number of features was included, this limits the applicability of the model we derived from this data set to the systems which make similar types of confusions. However, it is still interesting to investigate whether confusion probabilities provide additional information compared to standard evaluation metrics. We discuss how better coverage could be obtained in Section 6.

4.2 Regression Models

Table 5 shows the regression models we obtained using different feature sets. All models were obtained using stepwise linear regression, using the Akaike information criterion (AIC) for variable

[2]We also experimented with using % unclassified as an additional feature, since % of rejections is known to be a problem for spoken dialogue systems. However, it did not improve the models, and we do not report it here for brevity.

	Actual				
Predicted	contradictory	correct	irrelevant	non-content	pc_incomplete
contradictory	175	86	3	0	43
correct	25	752	1	4	26
irrelevant	31	12	16	4	29
non-content	1	3	2	95	3
pc_incomplete	200	317	40	28	419

Table 3: Confusion matrix for BEETLE II. System predicted values are in rows; actual values in columns.

selection implemented in the R stepwise regression library. As measures of model quality, we report R^2, the percentage of variance accounted for by the models (a typical measure of fit in regression modeling), and mean squared error (MSE). These were estimated using leave-one-out cross-validation, since our data set is small.

We used feature ablation to evaluate the contribution of different features. First, we investigated models using precision, recall or F-score alone. As can be seen from the table, precision is not predictive of learning gain, while F-score and recall perform similarly to one another, with $R^2 = 0.12$. In comparison, the model using only confusion frequencies has substantially higher estimated R^2 and a lower MSE.[3] In addition, out of the 3 confusion features, only one is selected as predictive. This supports our hypothesis that different types of errors may have different importance within a practical system.

The confusion frequency feature chosen by the stepwise model ("predicted-pc_incomplete-actual-contradictory") has a reasonable theoretical justification. Previous research shows that students who give more correct or partially correct answers, either in human-human or human-computer dialogue, exhibit higher learning gains, and this has been established for different systems and tutoring domains (Litman et al., 2009). Consequently, % of contradictory answers is negatively predictive of learning gain. It is reasonable to suppose, as predicted by our model, that systems that do not identify such answers well, and therefore do not remediate them correctly, will do worse in terms of learning outcomes.

Based on this initial finding, we investigated the models that combined either F scores or the full set of intrinsic evaluation scores with confusion frequencies. Note that if the full set of metrics (precision, recall, F score) is used, the model derives a more complex formula which covers about 33% of the variance. Our best models, however, combine the averaged scores with confusion frequencies, resulting in a higher R^2 and a lower MSE (22% relative decrease between the "scores.f" and "conf+scores.f" models in the table). This shows that these features have complementary information, and that combining them in an application-specific way may help to predict how the components will behave in practice.

5 Using prediction models in evaluation

The models from Table 5 can be used to compare different possible implementations of the interpretation component, under the assumption that the component with a higher predicted learning gain score is more appropriate to use in an ITS. To show how our predictive models can be used in making implementation decisions, we compare three possible choices for an interpretation component: the original BEETLE II interpreter, the baseline classifier described earlier, and a new decision tree classifier trained on our data.

We built a decision tree classifier using the Weka implementation of C4.5 pruned decision trees, with default parameters. As features, we used lexical similarity scores computed by the `Text::Similarity` package[4]. We computed 8 features: the similarity between student answer and either the expected answer text or the question text, using 4 different scores: raw number of overlapping words, F1 score, lesk score and cosine score. Its intrinsic evaluation scores are shown in Table 6, estimated using 10-fold cross-validation.

We can compare BEETLE II and baseline classifier using the "scores.all" model. The predicted

[3] The decrease in MSE is not statistically significant, possibly because of the small data set. However, since we observe the same pattern of results across our models, it is still useful to examine.

[4] http://search.cpan.org/dist/Text-Similarity/

Name	Variables
scores.fm	fmeasure.microaverage, fmeasure.macroaverage, fmeasure.correct, fmeasure.contradictory, fmeasure.pc_incomplete,fmeasure.non-content, fmeasure.irrelevant
scores.precision	precision.microaverage, precision.macroaverage, precision.correct, precision.contradictory, precision.pc_incomplete,precision.non-content, precision.irrelevant
scores.recall	recall.microaverage, recall.macroaverage, recall.correct, recall.contradictory, recall.pc_incomplete,recall.non-content, recall.irrelevant
scores.all	scores.fm + scores.precision + scores.recall
conf	Freq.predicted.contradictory.actual.correct, Freq.predicted.pc_incomplete.actual.correct, Freq.predicted.pc_incomplete.actual.contradictory

Table 4: Feature sets for regression models

Variables	Cross-validation R^2	Cross-validation MSE	Formula
scores.f	0.12 (0.02)	0.0232 (0.0302)	$0.32 + 0.56 * fmeasure.microaverage$
scores.precision	0.00 (0.00)	0.0242 (0.0370)	0.61
scores.recall	0.12 (0.02)	0.0232 (0.0310)	$0.37 + 0.56 * recall.microaverage$
conf	0.25 (0.03)	0.0197 (0.0262)	$0.74 - 0.56 * Freq.predicted.pc\_incomplete.actual.contradictory$
scores.all	0.33 (0.03)	0.0218 (0.0264)	$0.63 + 4.20 * fmeasure.microaverage - 1.30 * precision.microaverage - 2.79 * recall.microaverage - 0.07 * recall.non - content$
conf+scores.f	0.36 (0.03)	0.0179 (0.0281)	$0.52 - 0.66 * Freq.predicted.pc\_incomplete.actual.contradictory + 0.42 * fmeasure.correct - 0.07 * fmeasure.non - content$
full (conf+scores.all)	0.49 (0.02)	0.0189 (0.0248)	$0.88 - 0.68 * Freq.predicted.pc\_incomplete.actual.contradictory - 0.06 * precision.non\_domain + 0.28 * recall.correct - 0.79 * precision.microaverage + 0.65 * fmeasure.microaverage$

Table 5: Regression models for learning gain. R^2 and MSE estimated with leave-one-out cross-validation. Standard deviation in parentheses.

score for BEETLE II is 0.66. The predicted score for the baseline is 0.28. We cannot use the models based on confusion scores ("conf", "conf+scores.f" or "full") for evaluating the baseline, because the confusions it makes are always to predict that the answer is correct when the actual label is "incomplete" or "contradictory". Such situations were too rare in our training data, and therefore were not included in the models (as discussed in Section 4.1). Additional data will need to be collected before this model can reasonably predict baseline behavior.

Compared to our new classifier, BEETLE II has lower overall accuracy (0.43 vs. 0.53), but performs micro- and macro- averaged scores. BEETLE II precision is higher than that of the classifier. This is not unexpected given how the system was designed: since misunderstandings caused dialogue breakdown in pilot tests, the interpreter was built to prefer rejecting utterances as uninterpretable rather than assigning them to an incorrect class, leading to high precision but lower recall.

However, we can use all our predictive models to evaluate the classifier. We checked the the confusion matrix (not shown here due to space limitations), and saw that the classifier made some of the same types of confusions that BEETLE II interpreter made. On the "scores.all" model, the predicted learning gain score for the classifier is 0.63, also very close to BEETLE II. But with the "conf+scores.all" model, the predicted score is 0.89, compared to 0.59 for BEETLE II, indicating that we should prefer the newly built classifier.

Looking at individual class performance, the classifier performs better than the BEETLE II interpreter on identifying "correct" and "contradictory" answers, but does not do as well for partially correct but incomplete, and for irrelevant answers. Using our predictive performance metric highlights the differences between the classifiers and effectively helps determine which confusion types are the most important.

One limitation of this prediction, however, is that the original system's output is considerably more complex: the BEETLE II interpreter explicitly identifies correct, incorrect and missing parts of the student answer which are then used by the system to formulate adaptive feedback. This is an important feature of the system because it allows for implementation of strategies such as acknowledging and restating correct parts of the answer. However, we could still use a classifier to "double-check" the interpreter's output. If the predictions made by the original interpreter and the classifier differ, and in particular when the classifier assigns the "contradictory" label to an answer, BEETLE II may choose to use a generic strategy for contradictory utterances, e.g. telling the student that their answer is incorrect without specifying the exact problem, or asking them to re-read portions of the material.

Label	prec.	recall	F1
correct	0.66	0.76	0.71
pc_incomplete	0.38	0.34	0.36
contradictory	0.40	0.35	0.37
irrelevant	0.07	0.04	0.05
non-content	0.62	0.76	0.68
macroaverage	0.43	0.45	0.43
microaverage	0.51	0.53	0.52

Table 6: Intrinsic evaluation scores for our newly built classifier.

6 Discussion and Future Work

In this paper, we proposed an approach for cost-sensitive evaluation of language interpretation within practical applications. Our approach is based on the PARADISE methodology for dialogue system evaluation (Walker et al., 2000). We followed the typical pattern of a PARADISE study, but instead of relying on a variety of features that characterize the interaction, we used scores that reflect only the performance of the interpretation component. For BEETLE II we could build regression models that account for nearly 50% variance in the desired outcomes, on par with models reported in earlier PARADISE studies (Möller et al., 2007; Möller et al., 2008; Walker et al., 2000; Larsen, 2003). More importantly, we demonstrated that combining averaged scores with features based on confusion frequencies improves prediction quality and allows us to see differences between systems which are not obvious from the scores alone.

Previous work on task-based evaluation of NLP components used RTE or information extraction as target tasks (Sammons et al., 2010; Yuret et al., 2010; Miyao et al., 2008), based on standard corpora. We specifically targeted applications which involve human-computer interaction, where running task-based evaluations is particularly expen-

sive, and building a predictive model of system performance can simplify system development.

Our evaluation data limited the set of features that we could use in our models. For most confusion features, there were not enough instances in the data to build a model that would reliably predict learning gain for those cases. One way to solve this problem would be to conduct a user study in which the system simulates random errors appearing some of the time. This could provide the data needed for more accurate models.

The general pattern we observed in our data is that a model based on F-scores alone predicts only a small proportion of the variance. If a full set of metrics (including F-score, precision and recall) is used, linear regression derives a more complex equation, with different weights for precision and recall. Instead of the linear model, we may consider using a model based on F_β score, $F_\beta = (1+\beta^2)\frac{PR}{\beta^2 P+R}$, and fitting it to the data to derive the β weight rather than using the standard F_1 score. We plan to investigate this in the future.

Our method would apply to a wide range of systems. It can be used straightforwardly with many current spoken dialogue systems which rely on classifiers to support language understanding in domains such as call routing and technical support (Gupta et al., 2006; Acomb et al., 2007). We applied it to a system that outputs more complex logical forms, but we showed that we could simplify its output to a set of labels which still allowed us to make informed decisions. Similar simplifications could be derived for other systems based on domain-specific dialogue acts typically used in dialogue management. For slot-based systems, it may be useful to consider concept accuracy for recognizing individual slot values. Finally, for tutoring systems it is possible to annotate the answers on a more fine-grained level. Nielsen et al. (2008) proposed an annotation scheme based on the output of a dependency parser, and trained a classifier to identify individual dependencies as "expressed", "contradicted" or "unaddressed". Their system could be evaluated using the same approach.

The specific formulas we derived are not likely to be highly generalizable. It is a well-known limitation of PARADISE evaluations that models built based on one system often do not perform well when applied to different systems (Möller et al., 2008). But using them to compare implementation variants during the system development, without re-running user evaluations, can provide important information, as we illustrated with an example of evaluating a new classifier we built for our interpretation task. Moreover, the confusion frequency feature that our models picked is consistent with earlier results from a different tutoring domain (see Section 4.2). Thus, these models could provide a starting point when making system development choices, which can then be confirmed by user evaluations in new domains.

The models we built do not fully account for the variance in the training data. This is expected, since interpretation performance is not the only factor influencing the objective outcome: other factors, such choosing the the appropriate tutoring strategy, are also important. Similar models could be built for other system components to account for their contribution to the variance. Finally, we could consider using different learning algorithms. Möller et al. (2008) examined decision trees and neural networks in addition to multiple linear regression for predicting user satisfaction in spoken dialogue. They found that neural networks had the best prediction performance for their task. We plan to explore other learning algorithms for this task as part of our future work.

7 Conclusion

In this paper, we described an evaluation of an interpretation component of a tutorial dialogue system using predictive models that link intrinsic evaluation scores with learning outcomes. We showed that adding features based on confusion frequencies for individual classes significantly improves the prediction. This approach can be used to compare different implementations of language interpretation components, and to decide which option to use, based on the predicted improvement in a task-specific target outcome metric trained on previous evaluation data.

Acknowledgments

We thank Natalie Steinhauser, Gwendolyn Campbell, Charlie Scott, Simon Caine, Leanne Taylor, Katherine Harrison and Jonathan Kilgour for help with data collection and preparation; and Christopher Brew for helpful comments and discussion. This work has been supported in part by the US ONR award N000141010085.

References

Kate Acomb, Jonathan Bloom, Krishna Dayanidhi, Phillip Hunter, Peter Krogh, Esther Levin, and Roberto Pieraccini. 2007. Technical support dialog systems: Issues, problems, and solutions. In *Proceedings of the Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies*, pages 25–31, Rochester, NY, April.

Gwendolyn C. Campbell, Natalie B. Steinhauser, Myroslava O. Dzikovska, Johanna D. Moore, Charles B. Callaway, and Elaine Farrow. 2009. The DeMAND coding scheme: A "common language" for representing and analyzing student discourse. In *Proceedings of 14th International Conference on Artificial Intelligence in Education (AIED), poster session*, Brighton, UK, July.

Myroslava O. Dzikovska, Charles B. Callaway, Elaine Farrow, Johanna D. Moore, Natalie B. Steinhauser, and Gwendolyn E. Campbell. 2009. Dealing with interpretation errors in tutorial dialogue. In *Proceedings of the SIGDIAL 2009 Conference*, pages 38–45, London, UK, September.

Myroslava Dzikovska, Diana Bental, Johanna D. Moore, Natalie B. Steinhauser, Gwendolyn E. Campbell, Elaine Farrow, and Charles B. Callaway. 2010a. Intelligent tutoring with natural language support in the Beetle II system. In *Sustaining TEL: From Innovation to Learning and Practice - 5th European Conference on Technology Enhanced Learning, (EC-TEL 2010)*, Barcelona, Spain, October.

Myroslava O. Dzikovska, Johanna D. Moore, Natalie Steinhauser, Gwendolyn Campbell, Elaine Farrow, and Charles B. Callaway. 2010b. Beetle II: a system for tutoring and computational linguistics experimentation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-2010) demo session*, Uppsala, Sweden, July.

Kate Forbes-Riley and Diane J. Litman. 2006. Modelling user satisfaction and student learning in a spoken dialogue tutoring system with generic, tutoring, and user affect parameters. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL '06)*, pages 264–271, Stroudsburg, PA, USA.

Kate Forbes-Riley, Diane Litman, Amruta Purandare, Mihai Rotaru, and Joel Tetreault. 2007. Comparing linguistic features for modeling learning in computer tutoring. In *Proceedings of the 2007 conference on Artificial Intelligence in Education: Building Technology Rich Learning Contexts That Work*, pages 270–277, Amsterdam, The Netherlands. IOS Press.

Narendra K. Gupta, Gökhan Tür, Dilek Hakkani-Tür, Srinivas Bangalore, Giuseppe Riccardi, and Mazin Gilbert. 2006. The AT&T spoken language understanding system. *IEEE Transactions on Audio, Speech & Language Processing*, 14(1):213–222.

Pamela W. Jordan, Maxim Makatchev, and Umarani Pappuswamy. 2006. Understanding complex natural language explanations in tutorial applications. In *Proceedings of the Third Workshop on Scalable Natural Language Understanding*, ScaNaLU '06, pages 17–24.

Lars Bo Larsen. 2003. Issues in the evaluation of spoken dialogue systems using objective and subjective measures. In *Proceedings of the 2003 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 209–214.

David D. Lewis. 1991. Evaluating text categorization. In *Proceedings of the workshop on Speech and Natural Language*, HLT '91, pages 312–318, Stroudsburg, PA, USA.

Diane Litman, Johanna Moore, Myroslava Dzikovska, and Elaine Farrow. 2009. Using natural language processing to analyze tutorial dialogue corpora across domains and modalities. In *Proceedings of 14th International Conference on Artificial Intelligence in Education (AIED)*, Brighton, UK, July.

Yusuke Miyao, Rune Sætre, Kenji Sagae, Takuya Matsuzaki, and Jun'ichi Tsujii. 2008. Task-oriented evaluation of syntactic parsers and their representations. In *Proceedings of ACL-08: HLT*, pages 46–54, Columbus, Ohio, June.

Sebastian Möller, Paula Smeele, Heleen Boland, and Jan Krebber. 2007. Evaluating spoken dialogue systems according to de-facto standards: A case study. *Computer Speech & Language*, 21(1):26 – 53.

Sebastian Möller, Klaus-Peter Engelbrecht, and Robert Schleicher. 2008. Predicting the quality and usability of spoken dialogue services. *Speech Communication*, pages 730–744.

Rodney D. Nielsen, Wayne Ward, and James H. Martin. 2008. Learning to assess low-level conceptual understanding. In *Proceedings 21st International FLAIRS Conference*, Coconut Grove, Florida, May.

Mihai Rotaru and Diane J. Litman. 2006. Exploiting discourse structure for spoken dialogue performance analysis. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 85–93, Stroudsburg, PA, USA.

Mark Sammons, V.G.Vinod Vydiswaran, and Dan Roth. 2010. "Ask not what textual entailment can do for you...". In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1199–1208, Uppsala, Sweden, July.

Marilyn A. Walker, Candace A. Kamm, and Diane J. Litman. 2000. Towards Developing General Models of Usability with PARADISE. *Natural Language Engineering*, 6(3).

Deniz Yuret, Aydin Han, and Zehra Turgut. 2010. SemEval-2010 task 12: Parser evaluation using textual entailments. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 51–56, Uppsala, Sweden, July.