# Which System Differences Matter? Using $\ell_1/\ell_2$ Regularization to Compare Dialogue Systems

**José P. González-Brenes** and **Jack Mostow**
Project LISTEN
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{joseg,mostow}@cs.cmu.edu

## Abstract

We investigate how to jointly explain the performance and behavioral differences of two spoken dialogue systems. The Join Evaluation and Differences Identification (JEDI), finds differences between systems relevant to performance by formulating the problem as a multi-task feature selection question. JEDI provides evidence on the usefulness of a recent method, $\ell_1/\ell_p$-regularized regression (Obozinski et al., 2007). We evaluate against manually annotated success criteria from real users interacting with five different spoken user interfaces that give bus schedule information.

## 1 Introduction

This paper addresses the problem of how to determine which differences between two versions of a system affect their behavior. Researchers in Spoken Dialogue Systems (SDSs) can be perplexed as to which of the differences between alternative systems affect performance metrics (Bacchiani et al., 2008). For example, when testing on real users at different periods of time, the variance of the performance metrics might be higher than the difference between systems, causing (i) significantly different scores in identical systems deployed at different times, and (ii) the same score on different systems (González-Brenes et al., 2009).

We approach the problem of finding which system differences matter by describing dialogues as feature vectors constructed from the logs of dialogs generated by the SDSs interacting with real users. Hence, we aim to identify features that jointly characterize the system differences and the performance of the SDS being evaluated. These features should be able to (i) predict a performance metric and (ii) distinguish between the two SDS being evaluated.

The main contribution of this paper is a novel algorithm for detecting differences between two systems that can explain performance. Additionally, we provide details on how to implement state-of-the-art multi-task learning for SDSs.

The rest of this manuscript is organized as follows. Section 2 reviews multi-task feature selection. Section 3 describes two algorithms to find which system differences matter. Section 4 describes the specific SDS used to illustrate our algorithms. Section 5 presents some experimental results. Section 6 reviews related prior work. Section 7 presents some concluding remarks and future work. Appendix A provides implementation details of the multi-task learning approach we used.

## 2 Feature Selection

In this section we describe how we use regression to perform feature selection. Feature selection methods construct and select subsets of features in order to build a good predictor. We focus our attention on feature selection methods that use complexity (regularization) penalties, because of their recent theoretical and experimental success (Yuan and Lin, 2006; Park and Hastie, 2007). We provide a more rigorous description of how to implement this formulation as an optimization problem in Appendix A.

We use labels to encode the output we want to predict. For example, if our performance metric is binary, we label successful dialogues with a $+1$, and unsuccesful dialogues with a $-1$. Given a training set consisting of labeled dialogues, we want to learn



*Proceedings of the SIGDIAL 2011: the 12th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 8–17,
Portland, Oregon, June 17-18, 2011. 

a model that assigns a label to unseen dialogues. We follow an approach called empirical risk minimization (Obozinski et al., 2007), that aims to minimize the error of fitting the training data, while penalizing the complexity of the model:

$$\textbf{Minimize} \boxed{\text{Model loss}} + \lambda \boxed{\text{Complexity}} \quad (1)$$

Here the hyper-parameter $\lambda$ controls the trade-off between a better fit to the training data (with a higher risk of over-fitting it), and a simpler model, with fewer features selected (and less predictive power). We now review the two components of risk minimization, model loss and complexity penalty.

### 2.1 Model Loss

We model probabilistically the loss of our model against the real-life phenomenon studied. Given a dialogue $x$, with correct label $l$, its loss using a model $\boldsymbol{\beta}$ is:

$$\text{loss}_\beta(\hat{y}, x) \equiv P(y = l|x; \text{reality}) - P(\hat{y} = l|x; \boldsymbol{\beta}) \quad (2)$$

Here $\hat{y}$ is the predicted value of the event $y$. Since $l$ is the true label, $P(y = l|x; \text{reality}) = 1$. To get the overall loss of the model, we aggregate over the prediction loss of each of the dialogues in the training set by summing their individual loss calculated with Equation 2. Let $X = \{x^{(1)}, x^{(2)}, \ldots x^{(n)}\}$ be the $n$ dialogues in the training set. Then the overall loss of model $\boldsymbol{\beta}$ is:

$$\text{loss}_\beta(y^{(1)}, x^{(1)}) + \cdots + \text{loss}_\beta(y^{(n)}, x^{(n)})$$

Since we use discrete labels, we use a logistic function to model their probability. Let $x_1, \ldots x_k$ be the $k$ features extracted from dialogue $x$. Then the logistic regression model is:

$$P(\hat{y} = +1|x; \boldsymbol{\beta}) = \frac{1}{Z} \exp(\beta_1 x_1 + \cdots + \beta_k x_k)$$

Here $\beta_1 ... \beta_k$ are the parameters of the model, and $Z$ simply normalizes $P$ to ensure that $P$ is a valid probability function (the range of $P$ should be 0 to 1):

$$Z = 1 + \exp(\beta_1 x_1 + \cdots + \beta_k x_k)$$

Multi-task learning solves related regression problems at the same time using a shared representation. We now describe the risk-minimization formulation for multi-task learning. Let $y^m$ be the value of the performance metric. Let $y^s$ be the label of the system that generated the dialogue. The individual dialogue loss of using models $\boldsymbol{\beta}^m$ and $\boldsymbol{\beta}^s$ is:

$$\text{loss}_{\beta^m}(\hat{y}^m, x) + \text{loss}_{\beta^s}(\hat{y}^s, x)$$

### 2.2 Complexity Penalties

We consider a feature $x_i$ to be selected into the model if its regression coefficient $\beta_i$ is non-zero. Complexity penalties encourage selecting only a few features. We review several commonly used penalties (Zou and Hastie, 2005):

- **$\boldsymbol{\ell_2}$ Penalty.** Under some circumstances $\ell_2$ penalties perform better than other types of penalties (Zou and Hastie, 2005). The $\ell_2$ penalty for a model $\boldsymbol{\beta}$ is:

$$||\boldsymbol{\beta}||_{\ell_2} \equiv \sqrt{(\beta_1)^2 + \cdots + (\beta_k)^2}$$

- **$\boldsymbol{\ell_1}$ Penalty.** An $\ell_1$ penalty induces sparsity by setting *many* parameters of the model $\boldsymbol{\beta}$ to exactly zero (Tibshirani, 1996).

$$||\boldsymbol{\beta}||_{\ell_1} \equiv |\beta_1| + \cdots + |\beta_k|$$

- **$\boldsymbol{\ell_1/\ell_2}$ Penalty.** Yuan and Lin (2006) proposed a group penalty for penalizing groups of features simultaneously. Previous work has shown that grouping features between tasks encourages features to be used either by all tasks or by none (Turlach et al., 2005; Obozinski et al., 2007; Lounici et al., 2009; Puniyani et al., 2010). Our $\ell_1/\ell_2$ penalty is:

$$\left|\sqrt{(\beta_1^m)^2 + (\beta_1^s)^2}\right| + ... + \left|\sqrt{(\beta_k^m)^2 + (\beta_k^s)^2}\right|$$

## 3 Finding Features that Predict Performance and System Differences

We find system differences that are predictive of SDS performance, relying on:

- *Describing dialogues as feature vectors.* The behavior of the systems must be describable by features extracted from the logs of the systems. A discussion of feature engineering for dialogue systems is found in (González-Brenes and Mostow, 2011).

- *Finding system differences.* The features of a classifier that distinguishes between SDSs, can be used to identify their differences (González-Brenes et al., 2009). When comparing two SDSs, we label the baseline system with $-1$, and the alternate version with $+1$.

- *Modeling performance.* Although our approach does not depend on a specific performance metric, in this paper we use dialogue success, a binary indicator that triggers that the user's query was answered by the SDS. Task completion is cheaper to compute than dialogue success, as it does not require a manual human labeled reference, but we consider that dialogue success is a more accurate metric. Task completion is used in commercial applications (Bacchiani et al., 2008), and has been extensively studied in the literature (Walker et al., 2001; Walker et al., 2002; Hajdinjak and Mihelic, 2006; Levin and Pieraccini, 2006; Möller et al., 2007; Möller et al., 2008; Schmitt et al., 2010). We encode success of dialogues by manually annotating them with a binary variable that distinguishes if the user query is fulfilled by the SDS.

We now present two algorithms to find what differences matter between systems. We introduce Serial EvaluatioN Analysis (SERENA) as a scaffold for the Join Evaluation and Differences Identification (JEDI) algorithm.

### 3.1 SERENA algorithm

The input to SERENA is a collection of log files created by two different SDSs and two functions that represent the correct label for the regression tasks. In our case these functions should return binary labels $(+1, -1)$: one task distinguishes between successful and unsuccessful dialogues, and the other task distinguishes a baseline from an alternative SDS version. SERENA's objective is to select features from one task, and use them to predict the other task. For example, SERENA selects features that predict differences between versions, and uses them to predict performance.

Algorithm 1 provides the pseudo-code for SERENA. Line 1 builds the training set $\mathbf{X}$ from parsing the logs of the SDSs. Lines 2 and 3 create the output variables $y$ for the regression tasks. Line 6 returns the most predictive features using $\ell_1$ regularization as described in Section 2. Line 8 builds a new training set, removing the features that were not selected in line 6. Line 9 builds the final coefficients by fitting a $\ell_2$-regularized model using a constant $\lambda_c$. We calculate the coefficients using an $\ell_2$ penalty, because it has a better fit to the data (Zou and Hastie, 2005). Moreover, by using the same penalty, we control for the idiosyncrasies different penalties have in parameter learning. In the experiments described in Section 5, all of our experiments are reported fitting a $\ell_2$-regularized models.

**Algorithm 1** SERENA algorithm

**Require:** $\mathbf{Logs_1}, \mathbf{Logs_2}$ are the collections of SDS logs of two systems. $\mathbf{task_1}$, $\mathbf{task_2}$ are functions that return the value of a performance metric, and which system is being evaluated ($-1$ if is the baseline, $+1$ otherwise).

1: $\mathbf{X} \leftarrow \text{extract\_features}(\mathbf{Log}_1, \mathbf{Log}_2)$
2: $\boldsymbol{y}^{t_1} \leftarrow \begin{bmatrix} \text{task}_1(\mathbf{Logs}_1) \\ \text{task}_1(\mathbf{Logs}_2) \end{bmatrix}$
3: $\boldsymbol{y}^{t_2} \leftarrow \begin{bmatrix} \text{task}_2(\mathbf{Logs}_1) \\ \text{task}_2(\mathbf{Logs}_2) \end{bmatrix}$
4: // Select features that explain both tasks:
5: **for** $\lambda = \{0.1, 0.2, \dots\}$ **do**
6: $\quad \boldsymbol{\beta}^{\boldsymbol{t_1}} \leftarrow \text{regression}_{\ell_1}(\mathbf{X}, \boldsymbol{y}^{t_1}, \lambda)$
7: $\quad$ // Get feature weights:
8: $\quad \mathbf{X}' \leftarrow \mathbf{X}$; where $x_k | \forall x_k \in \mathbf{X}', \beta_k^{t_1} \neq 0$
9: $\quad \boldsymbol{\beta}^* \leftarrow \text{regression}_{\ell_2}(\mathbf{X}', \boldsymbol{y}^{t_2}, \lambda_c)$
10: **end for**
11: **return** $\boldsymbol{\beta}^*$

SERENA is not conmutative with regards to the order of the tasks: selecting the features that predict performance and using them to predict system differences is not the same as the reverse. More importantly, SERENA only searches in one of the tasks at a time. We are interested in finding the features that explain both tasks *simultaneously*. In the next subsection we describe JEDI which makes use of recent advances in multi-task feature selection in order to find the features for both tasks at the same time.

### 3.2 JEDI algorithm

Algorithm 2 provides the pseudo-code for JEDI. JEDI uses multi-task regression to find the features that affect performance and system differences

**Algorithm 2** JEDI algorithm

**Require:** $\mathbf{Logs_1}, \mathbf{Logs_2}$ are the collections of SDS logs of two systems. $\mathbf{task_1}$, $\mathbf{task_2}$ are functions that return the value of a performance metric, and which system is being evaluated ($-1$ if is the baseline, $+1$ otherwise).

$\mathbf{X} \leftarrow \text{extract\_features}(\mathbf{Log}_1, \mathbf{Log}_2)$
$\boldsymbol{y}^{t_1} \leftarrow \begin{bmatrix} \text{task}_1(\mathbf{Logs}_1) \\ \text{task}_1(\mathbf{Logs}_2) \end{bmatrix}$
$\boldsymbol{y}^{t_2} \leftarrow \begin{bmatrix} \text{task}_2(\mathbf{Logs}_1) \\ \text{task}_2(\mathbf{Logs}_2) \end{bmatrix}$
// Select features that explain both tasks:
**for** $\lambda = \{0.1, 0.2, \dots\}$ **do**
  $\boldsymbol{\beta^{t_1}}\boldsymbol{\beta^{t_2}} \leftarrow \text{regression}_{\ell_1/\ell_2}(\mathbf{X}, \vec{\mathbf{y}}^{t_1}, \vec{\mathbf{y}}^{t_2}, \lambda)$
  // Get feature weights:
  $\mathbf{X}' \leftarrow \mathbf{X}$; where $x_k | \forall x_k \in \mathbf{X}', \beta_k^{t_1} \neq 0$
  $\boldsymbol{\beta}^* \leftarrow \text{regression}_{\ell_2}(\mathbf{X}', \boldsymbol{y}^{t_2}, \lambda_c)$
**end for**
**return** $\boldsymbol{\beta}^*$

jointly. To ensure a sparse feature set across the tasks, it applies a $\ell_1/\ell_2$ penalty. In this way, a feature is either selected to predict performance and system differences, or it is not selected at all. In the next section we review a concrete example of a SDS in which we apply SERENA and JEDI.

## 4 Case Study

We use data collected from *Let's Go!*, an experimentary SDS available for research purposes[1] that provides bus planning information to the Pittsburgh East End community (Raux et al., 2006). Figure 1 shows a sample dialogue.

We assembled three disjoint datasets to have dialogues just before and after what we considered important architectural differences of Let's Go's implementation. By going through the change log, we realize that the magnitude of the differences varies widely. Table 1 describes the date the systems were deployed, the description of the change in the SDS, and the number of dialogues in each set. From the 86 dialogues of "Set A", half are a random sample of the logs before the acoustic model was retrained ($\mathbf{Log_1}$), and the other half is a sample after the acoustic model was retrained ($\mathbf{Log_2}$). In the case of the 84 dialogues of "Set B", half of the logs

[1] http://www.dialrc.org/

| | |
|---|---|
| **System:** | Welcome to the CMU Let's Go! Bus Information System. How may I help you? |
| **User:** | I'd like to go to Forbes and Murray. |
| **System:** | To Forbes and Murray. Where are you leaving from? |
| **User:** | Carnegie Mellon. |
| **System:** | From Carnegie Mellon. There are five buses running between Carnegie Mellon and Forbes and Murray: the 501, the 59U, the 61A, the 61B, and the 61C. What else can I do for you? |
| **User:** | When is the next one? |
| **System:** | The next 61C going to Forbes and Murray will leave Carnegie Mellon at 5:13 PM. |

Figure 1: Example of a *Let's Go* dialogue, from (Raux et al., 2006)

use an older version of the Olympus SDS infrastructure (Bohus et al., 2007), and the other half uses Olympus II. Since each system was deployed in a different period of time, we want to corroborate that we are modeling the differences among systems, and not seasonal. Hence, for control conditions, we also chose a data set that contained no major change to the system or to other conditions (Set C).

Sets were built by randomly sampling from the collection of logs. They have the same number of dialogues from each SDS version (baseline/alternate). Each dialogue was manually annotated to indicate whether the user's query was fulfilled, and we removed from our analysis the two dialogues that were only partially fulfilled. The number of successful dialogues is different from the number of unsuccessful dialogues.

We created a script to extract features from the log files of Let's Go!. The script has an explicit list of features to extract from the event logs, such as the words that were identified by the Automatic Speech Recognizer. Although this script is dependent on our specific log format, it should be a simple programming task to adapt it to a different dialogue system, provided its logs are comprehensive enough. The

Table 1: **Dataset Description**

| Set | Size | Description | Date | |
|---|---|---|---|---|
| A | 86 | Baseline | 8/05 | 10/05 |
| | | New acoustic model | 12/05 | 2/05 |
| B | 86 | Baseline | 8/06 | 10/06 |
| | | New SDS architecture | 6/07 | 7/07 |
| C | 84 | Baseline | 10/07 | 11/07 |
| | | No change | 11/07 | 12/07 |

script performs the standard transformation of centering feature values as $z$-scores with mean zero and standard deviation one.

Table 2 summarizes the properties we are interested to model. Dialogue properties are the features that summarize the behavior of the whole dialogue, and turn properties work at a finer-grain. We encode turn properties into features in the following way:

- **Global average.** Turn properties are averaged over the entire dialogue.
- **Beginning window.** Turn properties are averaged across an initial window. Based on preliminary experiments, we defined the window as the first 5 turns.
- **State.** We relied on the fact that SDSs are often engineered as finite state automata (Bohus et al., 2007). Properties are averaged across the states that belong to a specific dialogue state (for example, asking departure place). Because we are interested in early identification of differences, we restricted state features to be inside the beginning window.

## 5 Evaluation

We assess the performance of our algorithms by evaluating the classification accuracy using the features selected. To facilitate assessment of SDS, we only consider models that select up to 15 features. Figure 2 reports mean classification accuracy using five-fold cross-validation. Its first column describes how well the features selected perform on detecting system differences, and the second column describes how well they predict task success as a performance metric. We compare JEDI and SERENA against the following approaches:

Table 2: **Features**

| **Dialogue Properties** |
|---|
| # of re-prompted turns |
| # of turns |
| Mean Dialogue length |
| is evening?, is weekend?, 0-23 hour |
| **Turn Properties** |
| Occurrences of word $w$ |
| # of parse errors |
| # of unrecognized words |
| # of words |
| # of repeated words |
| # of unique words |
| Turn length |
| Words per minute |
| Failed prompts (number and percentage) |
| Mean Utterance Length |
| Barge-in (in seconds) |
| Machine-user pause (in seconds) |
| User-machine pause (in seconds) |
| Amplitude (power) statistics |

- **Majority classifier baseline.** A classifier that always selects the majority class (datasets B and C are not balanced in the number of successful dialogues).
- **Same Task Classifier** We report the classification accuracy of the model trained and tested on the *same* task. Features are selected using an $\ell_1$ penalty, and the coefficients are estimated with $\ell_2$-regularized logistic regression. For example, in the column of the left, SERENA uses the most predictive features of system differences to predict success, while the same task classifier uses them to predict system differences. The same task classifier does not answer "which system differences matter", it is just an interesting benchmark.

We used a one-sample $t$-test to check for statistically significant differences against the classification accuracy of the majority classifier baseline. We used a paired-sample $t$-test to check for significant differences in classification accuracy between classifiers. Paired samples have the same $\lambda$ hyper-parameter, which was described in the risk-

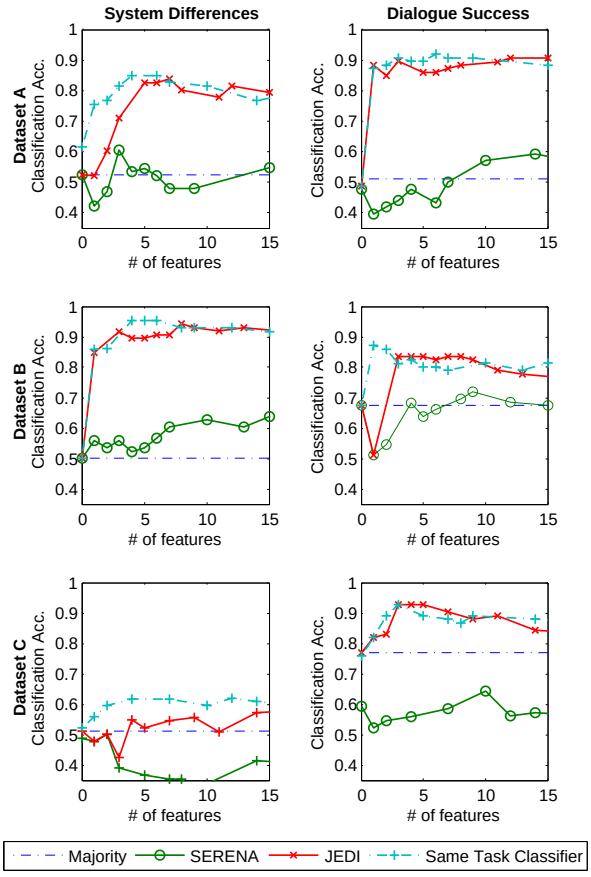


Figure 2: Classification accuracy of different feature selection techniques

minimization formulation explained in Section 2. This hyper-parameter is related to the number of features selected – as $\lambda$ increases, the number of features selected decreases. We use 5% as the significance level at which to reject the null hypothesis. When checking for statistical differences, we tested on the range of $\lambda$s computed[2].

First we investigate the performance of the simpler algorithm SERENA. For Dataset A, SERENA does not yield significant differences over the majority classifier baseline. For Dataset B, SERENA is significantly better than the majority classifier in predicting system differences, but is significantly worse for predicting success. This means that the order in which we choose the tasks in SERENA affects its performance. SERENA performs significantly worse in the Control Set C. We conclude that SERENA is not very reliable in predicting which system differences matter.

[2] $\lambda = \{100, 30, 25, 20, 19, 18, \ldots, 1, 0.5, 0.25, 0.1\}$

Table 3: **Features selected in Dataset A**

| Feature | Suc. | Diff. | JEDI |
|---|---|---|---|
| System-user pause | 5 | | 5 |
| Weekend night? | 3 | | |
| % of failed prompts | 4 | | |
| "Forbes_St." word | | 5 | 3 |
| User's max. power | | 5 | |

Table 4: **Features selected in Dataset B**

| Feature | Suc. | Diff. | JEDI |
|---|---|---|---|
| % of failed prompts | 5 | | 4 |
| User's power std.dev. | 5 | | |
| Weekend night? | 3 | | |
| Unrecognized word | | 5 | |
| Words/min. | | 4 | |
| User-system pause | | 5 | |
| Turn length | | 5 | 5 |

We now discuss how well JEDI is able to fill-in for the deficiencies of SERENA. As an "upper-bound", we will compare it to a classifier trained and tested in the same task. This classifier significantly dominates over the majority baseline, even for the the Control Set C, where there were no changes in the SDS. This suggests that the classifier might be picking up on seasonal differences. For Set A, JEDI performs significantly better than the majority classifier and than SERENA. For Set B, there are no significant differences between the upper-bound classifier and JEDI when predicting for changes in the SDS. Again, JEDI dominates over SERENA and the majority baseline. For the Control Set C, JEDI is not statistically different from the majority baseline. This is the expected behavior, since the difference in performance cannot be explained by the differences between the SDS. We hypothesize that the classification accuracy of JEDI could be used as a distance function between SDS: The closer the accuracy of distinguishing SDS is to 50%, the more similar the SDSs are. Conversely, when JEDI is able to classify system differences closer to 100%, it is because the SDSs are more different.

Tables 3 and 4 describe the features selected for Sets A and B respectively. The numbers indicate

in how many folds the feature was selected by JEDI and by classifiers trained to predict Success and SDS differences using five-fold cross validation. The $\lambda$ used is selected to contain the closest to five features (ties are resolved randomly). We only report features that appeared in at least three folds. In Dataset A we see that time of day is selected to predict dialogue success. Anecdotally, we have noticed that many users during weekend nights appear to be intoxicated when calling the system. JEDI does not select "is weekend night" as a feature, because it has little predictive power to detect system differences. In Dataset A, JEDI selects a speech recognition feature (the token "Forbes_St" was recognized), and an end-pointing feature. Since in Dataset A, the difference between systems correspond to a different acoustic model, these features make sense intuitively. In Dataset B, JEDI detected that the features most predictive with system differences and success are percentage of failed prompts and the length of the turn. The models for both systems make sense after the fact. However, neither model was known beforehand, nor did we know which of many features considered would turn out to be informative. Anecdotally, the documentation of the history of changes of Let's Go! is maintained manually. Sometimes, because of human error, this history is incomplete. The ability of JEDI to identify system differences has been able to help completing the history of changes (González-Brenes et al., 2009).

## 6 Relation to Prior Work

The scientific literature offers several performance metrics to assess SDS performance (Polifroni et al., 1992; Danieli and Gerbino, 1995; Bacchiani et al., 2008; Suendermann et al., 2010). SDS are evaluated using different objective and subjective metrics. Examples of objective metrics are the mean number of turns in the dialogue, and dialogue success. Subjective evaluations study measure satisfaction through controlled user studies. Ai et al. (2007) studied the differences in using assessment metrics with real users and paid users.

PARADISE, a notable example of a SDS subjective evaluation, finds linear predictors of a satisfaction score using automatic and hand-labeled features (Hajdinjak and Mihelic, 2006; Walker et al., 2001), or only automatic features (Hastie et al., 2002). Satisfaction scores are calibrated using surveys in controlled experiments (Möller et al., 2007; Möller et al., 2008). Alternatively, Eckert et al. (1998) proposed simulated users to evaluate SDSs. Their performance metric has to be tuned with a subjective evaluation as well, in which they refer to the PARADISE methodology. Our approach does not require user surveys to be calibrated. Moreover, it would be feasible to adapt JEDI to regress to PARADISE, or other performance metrics. Our work extends previous studies that define performance metrics, in proposing an algorithm that finds how system differences are related to performance.

## 7 Conclusions and Future Work

We have presented JEDI, a novel algorithm that finds features describing system differences relevant to a success metric. This is a novel, automated "glass box" assessment in the sense of linking changes in overall performance to specific behavioral changes. JEDI is an application of feature selection using regularized regression.

We have presented empirical evidence suggesting that JEDI's use of multi-task feature selection performs better than single-task feature selection. Future work could extend JEDI to quantify the variability in performance explained by the differences found. Common techniques in econometrics, such as the Seemingly Unrelated Regressions (SUR) formulation (Zellner, 1962), may prove useful for this.

In our approach we used a single binary evaluation criterion. By using a different loss function, JEDI can be extended to allow continuous-valued metrics. Moreover, previous work has argued that evaluating SDSs should not be based on just a single criterion (Paek, 2001). JEDI's multi-task formulation can be extended to include more than one performance criterion at the same time, and may prove helpful to understand trade-offs among different evaluation criteria.

## A Implementation Details of Feature Selection

In this appendix we review how to set-up multi-task feature selection as an optimization problem.

### A.1 $\ell_1$-Regularized Regression for Single-Task Feature Selection

We first review using regression with $\ell_1$ regularization for single-task feature selection. Given a training set represented by $\mathbf{X}$, denoting a $n \times k$ matrix, where $n$ is the number of dialogues, and $k$ is the number of features extracted for each dialogue, we want to find the coefficients of the parameter vector $\vec{\boldsymbol{\beta}}$, that can predict the output variables described in the vector $\vec{\mathbf{y}}$ of length $n$.

For this, we find the parameter vector that minimizes the loss function $J$, penalized by a regularization term (Tibshirani, 1996):

$$\underset{\vec{\boldsymbol{\beta}}}{\operatorname{argmin}} J(\mathbf{X}, \vec{\boldsymbol{\beta}}, \vec{\mathbf{y}}) + \lambda ||\vec{\boldsymbol{\beta}}||_{\ell_1} \tag{3}$$

In the case of binary classification, outputs are binary (any given $y = \pm 1$). A commonly used loss function $J$ is the Logistic Loss:

$$J_{\log}(x, \beta, y) \equiv \frac{1}{1 + e^{y(x \cdot \beta)}} \tag{4}$$

The $\ell_p$-norm of a vector $\vec{\boldsymbol{\beta}}$ is defined as:

$$||\vec{\boldsymbol{\beta}}||_{\ell_p} \equiv \big( \sum_{i=1}^{k} |\beta_i|^p \big)^{1/p}$$

The $\ell_\infty$-norm is defined as $||\vec{\boldsymbol{\beta}}||_{\ell_\infty} \equiv \max(\beta_1, \beta_2, \ldots, \beta_k)$.

The regularization term $||\vec{\boldsymbol{\beta}}||_{\ell_1}$ in Equation 3 controls model complexity: The higher the value of the hyper-parameter $\lambda$, the smaller number of features selected. Conversely, the smaller the value of $\lambda$, the better the fit to the training data, with higher risk of over-fitting it. Thus, Equation 3 jointly performs feature selection and parameter estimation; it induces sparsity by setting *many* coefficients of $\vec{\boldsymbol{\beta}}$ to zero (Tibshirani, 1996). Features with non-zero coefficients are considered the features selected.

### A.2 $\ell_1$-Regularized Regression for Multi-Task Feature Selection

$\ell_1$ regularization can be used to learn a classifier for each of $T$ prediction task *independently*. In our case we are interested in only two prediction tasks: version and success. We will index tasks with superscript $t$, and we define $\mathbf{X^t}$ as the $n \times k$ training data for task $t$, used to predict the output variable $\vec{\mathbf{y}}^t$. Learning each model separately yields the following optimization problem (Obozinski et al., 2007):

$$\underset{\vec{\boldsymbol{\beta}}^t}{\operatorname{argmin}} \sum_{t=1}^{T} J(\mathbf{X^t}, \vec{\boldsymbol{\beta}}^t, \vec{\mathbf{y}}^t) + \lambda ||\vec{\boldsymbol{\beta}}^t||_{\ell_1} \tag{5}$$

Solving this problem leads to individual sparsity in each task (each $\vec{\boldsymbol{\beta}}^t$ has many zeros), but the model does not enforce a common subset of features for all of the related output variables simultaneously (Turlach et al., 2005). In the next subsection we study how to achieve global sparsity across tasks.

### A.3 $\ell_1/\ell_p$-Regularized Regression for Multi-task Feature Selection

Although $\ell_1$-regularization is very successful at selecting individual features, it does not perform adequately when a group of features should enter or leave the model simultaneously (Yuan and Lin, 2006). Group LASSO (Yuan and Lin, 2006), which relies on $\ell_1/\ell_p$-regularization to overcome this limitation, by allowing groups of feature entering or leaving the model simultaneously. $\ell_1/\ell_p$ regularization has been studied for multi-task learning by grouping each of the $k$ features across the $T$ learning tasks (Turlach et al., 2005; Obozinski et al., 2007; Lounici et al., 2009; Puniyani et al., 2010).

Let us define $\mathbf{B}$ as a $n \times T$ matrix, whose $t^{th}$ column is the parameter vector for the task $t$. For example, since we have two tasks $\mathbf{B} = [\vec{\boldsymbol{\beta}}^{t=1}, \vec{\boldsymbol{\beta}}^{t=2}]$. Let $\vec{\boldsymbol{\beta}}_g$ denote the $g^{th}$ row of $\mathbf{B}$. In the context of multi-task learning, the $\ell_1/\ell_p$-norm of a matrix $\mathbf{B}$ is defined as (Obozinski et al., 2007; Puniyani et al., 2010):

$$||\mathbf{B}||_{\ell_1/\ell_p} \equiv \sum_{g=1}^{k} ||\vec{\boldsymbol{\beta}}_g||_{\ell_p} \tag{6}$$

Multi-task feature selection with $\ell_1/\ell_p$ regularization is formulated as (Obozinski et al., 2007; Puniyani et al., 2010):

$$\underset{\mathbf{B}}{\operatorname{argmin}} \sum_{t=1}^{T} J(\mathbf{X}^t, \vec{\boldsymbol{\beta}}^t, \vec{\mathbf{y}}^t) + \lambda ||\mathbf{B}||_{\ell_1/\ell_2} \tag{7}$$

When $T = 1$, the multi-task problem of Equation 7 reduces to the single-task problem of Equation 5.

### A.4 Optimization procedure

Puniyani et al. (2010) describe that finding the parameter coefficients $\mathbf{B}$ of Equation 7 can be achieved more easily by transforming the problem into an equivalent single-task multivariate regression. We follow their procedure to create $\vec{\boldsymbol{y}_{\boldsymbol{g}}}$, $\vec{\boldsymbol{\beta}_{\boldsymbol{g}}}$ and $\mathbf{X_g}$:

1. Concatenate the vectors $\vec{\mathbf{y}}^t$'s into a single vector $\vec{\mathbf{y}}_g$ of length $n \times T$. In our case, since we have only two tasks ($T = 2$), we get the vector $\vec{\mathbf{y}_\mathbf{g}} = \left[\begin{smallmatrix} \vec{\mathbf{y}}^{t=1} \\ \vec{\mathbf{y}}^{t=2} \end{smallmatrix}\right]$.

2. Similarly, we concatenate the $\vec{\boldsymbol{\beta}}^t$'s into a $k \times T$ vector $\vec{\boldsymbol{\beta}_{\boldsymbol{g}}}$, in our case $\vec{\boldsymbol{\beta}_{\boldsymbol{g}}} = \left[\begin{smallmatrix} \vec{\boldsymbol{\beta}}^{t=1} \\ \vec{\boldsymbol{\beta}}^{t=2} \end{smallmatrix}\right]$.

3. Build a $(n \cdot T) \times (k \cdot T)$ block-diagonal matrix $\mathbf{X_g}$, where $\mathbf{X}^t$'s are placed along the diagonal, and the rest of the elements are set to zero. In our case since we only have two tasks this is $\mathbf{X_g} = \left[\begin{smallmatrix} \mathbf{X^{t=1}} & \emptyset \\ \emptyset & \mathbf{X^{t=2}} \end{smallmatrix}\right]$, where each $\emptyset$ denotes a $n \times k$ zero-matrix. The expanded notation of $\mathbf{X_g}$ is:

$$\mathbf{X_g} \equiv \begin{bmatrix} \mathrm{x}^{\mathrm{t=1}}{}_1^{(1)} & \dots & \mathrm{x}^{\mathrm{t=1}}{}_k^{(1)} & 0 & \dots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ \mathrm{x}^{\mathrm{t=1}}{}_1^{(n)} & \dots & \mathrm{x}^{\mathrm{t=1}}{}_k^{(n)} & 0 & \dots & 0 \\ 0 & \dots & 0 & \mathrm{x}^{\mathrm{t=2}}{}_1^{(1)} & \dots & \mathrm{x}^{\mathrm{t=2}}{}_k^{(1)} \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & \mathrm{x}^{\mathrm{t=2}}{}_1^{(n)} & \dots & \mathrm{x}^{\mathrm{t=2}}{}_k^{(n)} \end{bmatrix}$$

Thus, the multi-task learning problem from Equation 7 is equivalent to (Yuan and Lin, 2006; Puniyani et al., 2010):

$$\underset{\mathbf{B}}{\operatorname{argmin}}\, J(\mathbf{X_g}, \vec{\boldsymbol{\beta}_{\boldsymbol{g}}}, \vec{\mathbf{y}_\mathbf{g}}) + \lambda ||\mathbf{B}||_{\ell_1/\ell_2} \qquad (8)$$

In this work we solve this optimization problem using an existing[3] implementation of Block Coordinate Descent (Schmidt et al., 2008) that solves regression problems with a $\ell_1/\ell_p$ penalty.

## Acknowledgments

This work was supported by the Institute of Education Sciences, U.S. Department of Education, through Grant R305A080628 to Carnegie Mellon University. The opinions expressed are those of the authors and do not necessarily represent the views of the Institute or U.S. Department of Education. We thank the educators, students, and LISTENers who helped generate, collect, and analyze our data, and the reviewers for their helpful comments. The first author was partially supported by the Costa Rican Ministry of Science and Technology (MICIT).

[3] Source code: `http://www.cs.ubc.ca/~murphyk/Software/L1CRF/`

## References

H. Ai, A. Raux, D. Bohus, M. Eskenazi, and D. Litman. 2007. Comparing spoken dialog corpora collected with recruited subjects versus real users. In *Proc. of the 8th SIGdial workshop on Discourse and Dialogue*.

M. Bacchiani, F. Beaufays, J. Schalkwyk, M. Schuster, and B. Strope. 2008. Deploying GOOG-411: Early lessons in data, measurement, and testing. In *IEEE International Conference on Acoustics, Speech and Signal Processing, 2008. ICASSP 2008*, pages 5260–5263.

D. Bohus, A. Raux, T. Harris, M. Eskenazi, and A. Rudnicky. 2007. Olympus: an open-source framework for conversational spoken language interface research. In *HLT-NAACL 2007 Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technology*.

M. Danieli and E. Gerbino. 1995. Metrics for evaluating dialogue strategies in a spoken language system. In *Proceedings of the 1995 AAAI Spring Symposium on Empirical Methods in Discourse Interpretation and Generation*, pages 34–39.

W. Eckert, E. Levin, and R. Pieraccini. 1998. Automatic evaluation of spoken dialogue systems. *TWLT13: Formal semantics and pragmatics of dialogue*, pages 99–110.

J. P. González-Brenes and J. Mostow. 2011. Classifying dialogue in high-dimensional space. *Transactions of Speech and Language Processing; Special Issue on Machine Learning for Robust and Adaptive Spoken Dialogue Systems*. In press.

J. P. González-Brenes, A. W. Black, and M. Eskenazi. 2009. Describing Spoken Dialogue Systems Differences. In *International Workshop on Spoken Dialogue Systems*, Irsee, Germany. Springer–Verlat.

M. Hajdinjak and F. Mihelic. 2006. The PARADISE evaluation framework: Issues and findings. *Computational Linguistics*, 32(2):263–272.

H. W. Hastie, R. Prasad, and M. Walker. 2002. Automatic evaluation: Using a date dialogue act tagger for

user satisfaction and task completion prediction. In *In LREC 2002*, pages 641–648.

E. Levin and R. Pieraccini. 2006. Value-based optimal decision for dialog systems. In *Spoken Language Technology Workshop, 2006. IEEE*, pages 198 –201.

K. Lounici, A.B. Tsybakov, M. Pontil, and van de Geer. 2009. Taking advantage of sparsity in multi-task learning. In *Conference on Learning Theory*, volume 1050, page 9, Montreal, Quebec.

S. Möller, P. Smeele, H. Boland, and J. Krebber. 2007. Evaluating spoken dialogue systems according to de-facto standards: A case study. *Computer Speech and Language*, 21(1):26 – 53.

S. Möller, K.P. Engelbrecht, and R. Schleicher. 2008. Predicting the quality and usability of spoken dialogue services. *Speech Communication*, 50(8-9):730–744.

G. Obozinski, B. Taskar, and M.I. Jordan. 2007. Multi-task feature selection. In *The Workshop of Structural Knowledge Transfer for Machine Learning in the 23rd International Conference on Machine Learning (ICML)*, Pittsburgh, PA.

T. Paek. 2001. Empirical methods for evaluating dialog systems. In *ACL 2001 Workshop on Evaluation Methodologies for Language and Dialogue systems*, pages 3–10.

M.Y. Park and T. Hastie. 2007. L1-regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(19):659–677.

J. Polifroni, L. Hirschman, S. Seneff, and V. Zue. 1992. Experiments in evaluating interactive spoken language systems. In *Proceedings of the workshop on Speech and Natural Language*, pages 28–33. Association for Computational Linguistics.

K. Puniyani, S. Kim, and E.P. Xing. 2010. Multi-population GWA mapping via multi-task regularized regression. *Bioinformatics*, 26(12):208.

A. Raux, D. Bohus, B. Langner, A.W. Black, and M. Eskenazi. 2006. Doing research on a deployed spoken dialogue system: one year of Let's Go! experience. In *Ninth International Conference on Spoken Language Processing*. ISCA.

M. Schmidt, K. Murphy, G. Fung, and R. Rosales. 2008. Structure learning in random fields for heart motion abnormality detection. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1 –8.

A. Schmitt, M. Scholz, W. Minker, J. Liscombe, and D. Suendermann. 2010. Is it possible to predict task completion in automated troubleshooters? In *INTERSPEECH*, pages 94–97.

D. Suendermann, J. Liscombe, R. Pieraccini, and K. Evanini. 2010. "How am I Doing?": A new framework to effectively measure the performance of automated customer care contact centers. In A. Neustein, editor, *Advances in Speech Recognition: Mobile Environments, Call Centers, and Clinics*, pages 155–180. Springer.

R. Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288.

B.A. Turlach, W.N. Venables, and S.J. Wright. 2005. Simultaneous variable selection. *Technometrics*, 47(3):349–363.

M. Walker, C. Kamm, and D. Litman. 2001. Towards developing general models of usability with PARADISE. *Natural Language Engineering*, 6(3):363–377.

M. A. Walker, I. Langkilde-Geary, H. W. Hastie, J. Wright, and A. Gorin. 2002. Automatically training a problematic dialogue predictor for a spoken dialogue system. *Journal of Artificial Intelligence Research*, 16:293–319.

M. Yuan and Y. Lin. 2006. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67.

A. Zellner. 1962. An efficient method of estimating seemingly unrelated regressions and tests for aggregation bias. *Journal of the American Statistical Association*, 57(298):pp. 348–368.

H. Zou and T. Hastie. 2005. Regularization and variable selection via the Elastic Net. *Journal of the Royal Statistical Society*, 67:301–320.