Describing Spoken Dialogue Systems Differences

José P. González-Brenes, Alan W Black, and Maxine Eskenazi

Language Technologies Institute,
Carnegie Mellon University, Pittsburgh PA 15213, USA
{joseg,awb,max}@cs.cmu.edu

Abstract. Conventional metrics in evaluating task-oriented Spoken Dialogue Systems (SDSs) are objective metrics such as the mean number of turns, or the estimated success rate. We provide further empirical evidence that these metrics are unreliable to distinguish across SDS: they fail to distinguish changes in the system, and detect differences where there are none. For instance, we report that the mean estimated success can have statistically significant differences (at the 5% significance level) on *exactly* the same system, over different, contiguous periods of time. We propose Dialogue System Difference Finder (DSDF), a novel model which can explain that the differences found using conventional metrics are due to seasonal and usage characteristics. DSDF is able to describe differences between multiple SDS, and it is different from traditional performance metrics used in the SDS community, in that it is more sensitive to changes among systems, has lower variance, and can also explain what the differences consist of. We used this model to find unexpected changes in our data sets. We believe that this lays the groundwork toward building a fully-automatic metric.

1 Introduction

We present a novel distance metric that is able to find and describe the differences across multiple Spoken Dialogue Systems (SDSs). When modifying a baseline SDS, researchers face the problem that individual changes may not reflect a given performance metric, and yet, a combination of these changes together may have a significant impact on the performance metric [Bacchiani et al. 2008, Pieraccini 2008]. This presents an enormous problem for experimentation, since the SDS designer has no means of knowing whether an incremental change in the system can lead to improvement. For experimentation, a finer grained metric is needed.

Describing the differences among SDSs can be thought of as a fundamental step forward toward a finer evaluation scheme, that can be useful for SDS researchers. This is especially useful when comparing different "black box" SDS that are meant to perform the same task.

We propose Dialogue System Difference Finder (DSDF), a model based on maximum entropy that is able to describe differences between multiple SDS. This model is different from traditional performance metrics used in the SDS community, in that it is more sensitive to changes among systems, has lower

variance, and can also explain what the differences consist of. We believe that this work lays the groundwork toward building a fully-automatic metric.

This paper is organized as follows: In Section 2 we provide a summary of relevant work and describe the statistical preliminaries on which we base our work. In Section 3 we discuss the model we are proposing to detect differences among SDS. We discuss the methodology to empirically validate our approach, and we provide the results of our observations in Section 4. Finally in Section 5, we conclude and address how we will further extend this work.

2 Background

Conventional metrics in evaluating task-oriented SDSs are objective metrics such as the mean number of turns, or the estimated success rate. [Bacchiani et al. 2008, Raux et al. 2006, Paek 2001]. Estimated success is a binary indicator that signals whether the user received information in the case of an information-giving SDS. The use of multiple individual objective metrics may provide contradictory criteria in selecting the best dialogue system [Paek 2001, Kamm et al. 1999]. Furthermore in section 4.3, we show that objective metrics are not sensitive to the differences in our dialogue corpus.

As an alternative to objective metrics, subjective evaluations measure user satisfaction through user studies. This approach is very expensive and needs a large number of users to interact with the system. Furthermore, there is evidence suggesting that laboratory user experimentation may not generalize to real users, and thus present low external validity [Ai et al. 2007].

Trying to offset some of these deficiencies, PARADISE [Hajdinjak and Mihelic 2006, Walker et al. 2001] was proposed as a general framework to measure SDS performance as a linear combination of objective and subjective measures using automatic and hand-labeled features. Hastie et al. [2002] proposed an extension to PARADISE, in which all features are extracted automatically, relying on an automatic dialogue act tagger and calculating a performance score with a regression tree. Both of these approaches require data to be collected via controlled experiments where users complete a satisfaction survey. A comparison of alternative satisfaction surveys and alternate ways on how to use machine learning techniques to regress those scores can be found in [Möller et al. 2008; 2007]. Eckert et al. [1998] propose simulated users to evaluate new SDS. However, their performance metric is a weighted sum of cost functions, and to get the initial weight of these parameters, they refer to the PARADISE methodology, and thus, it might also be suceptible to the paid users vs. real user dilemma described in Ai et al. [2007].

Paek [2001] gives a more detailed discussion of subjective performance metrics and objective metrics. He argues that instead of focusing on developing new metrics, new methodologies that allow comparative judgment are needed. Our work can be considered to address this comparative methodology paradigm.

To our knowledge, our distance function is a novel approach function that can be used for describing the differences between systems. In the rest of this

section we review the preliminary statistical knowledge on which we base our work, particularly we focus on maximum entropy classification (Section 2.1), and variable selection (Section 2.2).

2.1 Maximum Entropy Classification

Nigam et al. [1999] presents maximum entropy classification models as the posterior probability of a class label c, as:

$$p(c|x;\theta) = \frac{1}{Z(x)} \sum_{i}^{n} \exp(\theta_i f_i(x)) \tag{1}$$

Where f are feature functions $f_1, ..., f_n$ such that $f_i : X \to \mathbb{R}$, x is the input data, and Z is a normalization factor, that ensures the probability function is well-defined:

$$Z(x) = \sum_{c} \exp(\sum_{i} \theta_i f_i(x)) \tag{2}$$

Nigam et al. [1999] provide a review of successful usage of maximum entropy in language technologies. However, since we are interested in being able to interpret the features that are most relevant in our model, we need to perform variable selection.

2.2 Variable Selection

In the context of statistical machine learning, variable selection is equivalent to distinguishing between predictive and non-predictive features [Zhang and Huang 2008]. In most (unregularized) models, the number of training examples needed to learn the parameters grows as a function of the number of features [Ng 2004]. A larger number of parameters allows better fitting to the training data, and increases the chances of over-fitting. Variable selection helps to avoid over-fitting, and it has the collateral effect that it helps the "data miner" to have a better understanding of the process that generated the data [Guyon and Elisseeff 2003].

In maximum entropy models, variable selection can be achieved by penalizing the parameter vector with a regularization term, R. As described in Ng [2004], if we have m examples drawned i.i.d., we train θ to maximize the posterior log-probability:

$$\underset{\theta}{\operatorname{argmax}} \sum_{i}^{m} \log p(c^{(i)}|x^{(i)};\theta) - \lambda R(\theta) \tag{3}$$

We can set R to penalize on the number of features, by defining it as the L_p norm of θ:

$$R(\theta) = ||\theta||_p := (\sum_{i=0}^{n} |\theta_i|^p)^{1/p} \tag{4}$$

In equation (3), λ is the hyper-parameter that controls the trade-off between bias and variance introduced by the regularization term R: the higher the value

of λ, the more bias we introduce to the model, and we get a smaller feature space. Conversely, the smaller the value of λ, the better we fit the training data. The particular case of $\lambda = 0$, is a model without variable selection.

In an ideal scenario, we are interested in using the L_0 norm, which would effectively penalize the number of parameters of the model. Unfortunately, obtaining the optimal subset of features for a model is known to be an NP-hard problem. At this point we have two options for variable selection: (i) we can approximate the optimal value of the L_0 norm by performing search in the (combinatorial space) of features subspace, or (ii) use a different regularization term to shrink the parameters toward zero [Bishop 1995, Xing et al. 2001]. Although technically both approaches perform feature selection, some authors distinguish among them by exclusively referring to the subset selection (search methods) as feature selection, and designating the regularization technique as model selection or parameter shrinking. For a more in-depth description of the subset selection methods see [Guyon and Elisseeff 2003].

Although Xing et al. [2001] provide empirical evidence that subset selection methods may give better results than model selection in genomic microarray data, Zhang and Huang [2008] claim that model selection methods are more stable and that these methods are more computationally efficient for high-dimensional data. For instance, maximum entropy with L_1 or L_2 regularization only requires the solution of a convex optimization problem [Ng 2004].

It is worth noting that L_1 regularization is equivalent to imposing a Laplace prior to the parameter weights, and L_2 regularization is equivalent to imposing a Gaussian prior. Because of this, L_1 regularization is not differentiable for every point; this difficulty is offset by the theoretical results of Ng [2004], proving that L_1 regularization is less prone to over-fitting than the L_2 on the same amount of data. The problem of variable selection is very related to learning a structure in a graphical model [Lee et al. 2007].

3 Dialogue System Difference Finder (DSDF)

We use a maximum entropy model with L_1 regularization, as a distance function. Our interpretation is that classification accuracy between the output of two SDS defines the distance between them: The stronger the differences between the systems, the closer the classification accuracy should be to 100%. Conversely, the more similar the systems are, the lower the accuracy, as the systems become indistinguishable. We found that this interpretation of likelihood as distance function was also used in the context of face recognition in [Colmenarez and Huang 1996].

In addition to classification accuracy to provide the magnitude of distance, we also considered the Kullback-Leibler (KL) divergence (sometimes called cross-entropy), and the area of the probability mass assignment (using dot product). For improved clarity, since KL divergence is not symmetric, and the dot product is not a standard metric in the literature to evaluate classifiers, we report our results in the more commonly used metric of classification accuracy. However,

it is very important to note that in our context higher classification accuracy is not always desirable, since some of the data sets actually do not present any differences, as they are used as controls.

For our maximum entropy classification model, we standardize the value of the features to have mean zero by calculating their z-score. Hence, the value of each of the parameter θ from Equation 1 is proportional to the contribution of each feature. We use McCallum [2002]'s open-source implementation of L_1-regularized maximum entropy, version 2.0, which we modified to allow real-valued features.

3.1 Features

We are interested in automatically extracted features. This contrasts with previous work such as [Möller et al. 2008], that uses features that rely on manual annotation, like word error rate statistics. Table 1 summarizes the feature templates implemented.

The actual implementation of the feature templates takes four instantiations: global, beginning (window) of dialogue, and dialogue state. These instantiations represent from what part of the dialogue the feature is being extracted. For instance, the feature template "words recognized" expands into all the words that the recognizer is able to comprehend, and hence, it instantiates into features such as "how many times the word 'help' is recognized at the beginning of the dialogue". The number of turns that are considered in the window is specified as a parameter, currently set as 4, since it gave adequate results in preliminary experiments. Dialogue state features are extracted in the window, but further restrict the feature to being active only in certain dialogue states: "For dialogue state X, what is the mean utterance length?".

Table 1. Feature Templates

General	Lexical / Recoginition	Acoustic
# of re-prompted turns	# of times word W is recognized	Pause machine-user
% of re-prompted turns	# of words	Gap user-machine
Mean Dialogue length	# of repeated words	
# of turns	# of unique words	
# of failed prompts	Mean Utterance Length	
# of parse errors	Barge-in	
Dialogue state visited		
Outcome of the dialogue		

The "outcome of the dialogue" template expands to features that trigger whether the user's query was an uncovered route, uncovered neighborhood, uncovered place, or if results were informed. If any of these feature triggers, so

does the success feature. We avoid expanding the feature template of individual words globally. All of the others features are averaged globally, in the window, and by dialogue state. Dialogue states are "Arrival Place", "Greeting", "Bus Number or Departure Place", "Confirm", "Next Bus", "Query error", "Departure Neighborhood", "Departure Stop in Neighborhood", "Exact Travel Time" and "Other".

Maximum entropy classifiers are able to distinguish among classes by detecting differences in the mean value of the features. To detect changes in the standard deviation, a common technique is to include the value of the squared value of the features. We follow this standard practice.

4 Experiments

4.1 Data set

We have chosen nine sets of data collected from the Let's Go Public platform [Raux et al. 2006]. Let's Go is an SDS that provides bus planning information to the Pittsburgh East End community. It has collected over 80,000 dialogues since 2005, and its data is freely available for research purposes upon request. We selected a total of 11,736 dialogues of over five turn turns in length, evenly distributed among nine different data sets. Table 2 shows an explanation of the time line and the description of each data set.

We wanted to find sets of data that occurred just before and after what we considered major changes to the system. For this we went over the full log of changes to Let's Go. To have some control conditions, we also chose a data set that contained no major change to the system or to other conditions (sets 5-6 and sets 7-8 of Table 2). Interestingly, we had to revise Table 2 after running our model, since it found differences between changes that we initially considered to be minor.

Table 3 provides statistics on the number of turns and estimated success on our dialogue corpus. The high standard deviation in the number of turns is due to the fact that this metric is not normally distributed – it has a long tail resulting from people who stay on the line longer. In Section 4.3 we review how these conventional objective performance metrics are not reliable to detect changes across the data sets.

4.2 Experimental Design

We first provide further evidence on the deficiencies of conventional performance metrics to detect changes in SDS. We do this by reviewing the statistically significant difference between the number of turns and the estimated success across the data sets presented in Table 3.

We validate our model with two different experiments, both trained on the same training data sets, using 10-fold cross validation with a validation set (10% of the data) and a testing data set (10% of the data). We inspected $\lambda = [1.0, 10.0]$ in 1.0 increments and $\lambda = [10.0, 100.0]$ in 10.0 increments.

Table 2. Description of the data set

Name	Start Date	End Date	Description of Major Events
Set 1	2005-03-04	2005-04-28	System went live to the Pittsburgh community.
Set 2	2005-08-01	2005-09-30	Prompt changed to generic: "What can I do for you?"
Set 3	2005-12-01	2006-01-31	Acoustic model retrained.
Set 4	2006-08-01	2006-09-30	Language model retrained.
Set 5	2007-06-01	2007-06-30	End pointing experiment. First public version of Let's Go based on Olympus II.
Set 6	2007-07-01	2007-07-31	No system change from data set 5. Crashes reported.
Set 7	2007-10-20	2007-11-22	Tuned modem's acoustic settings, set random end pointing thresholds.
Set 8	2007-11-23	2007-12-31	Switch back to fixed endpointing thresholds. No major difference from data set 7.
Set 9	2008-12-01	2009-01-23	New end pointing strategy changed. SUNY Stony Brook experiment on user. strategies when confronted by system errors

Table 3. Description of the data set

Dataset	Turns Avg.	Turns Std. dev.	Success Avg.	Success Std. dev.
Set 1	17.1035	13.1994	58.28%	0.49
Set 2	16.9648	12.4947	61.69%	0.49
Set 3	18.3257	13.3877	54.79%	0.50
Set 4	16.7807	12.8700	74.08%	0.44
Set 5	16.9088	13.1077	76.32%	0.43
Set 6	17.5295	20.0300	71.65%	0.45
Set 7	15.7571	12.6864	78.62%	0.41
Set 8	16.4176	13.0109	78.54%	0.41
Set 9	14.6552	11.4973	74.33%	0.43

For the first experiment we attempt to find the best model that explains the differences across the data sets, and we chose the regularization hyper parameter that optimizes the classification using the validation set, and report the values of the testing set. In the second experiment we focus on explaining what the actual differences between the systems are, and thus we aim to obtain a sparse model that is "human readable". Since in Experiment 1, we inspected the data using different regularization hyper-parameters, we report the sparse model using the validation data set, maintaining the 'purity' of the testing set. We look for models that resulted in less than 15 features.

For both experiments we run binary maximum entropy classifiers, using consecutive datasets. Although we initially considered using multi-class maximum entropy classification, some of the classes parameters would become sparse (turn to zero) while others remain dense.

We hypothesize that the magnitude of the difference between the sets that do not present changes at all (sets 5-6 and 7-8) should be lower than the other ones (see Table 2).

4.3 Experimental Results

Table 4 summarizes the results our experiments. The column 'Set' describes the data sets compared, with the ones in which no major change was made to the system are marked with a star. For the best model and sparse models, we report the mean distance across folds between the sets (which is the classification accuracy), standard deviation, standard error and mean number of features selected. As a baseline, we use the number of turns and estimated success from Table 3. For the baseline we report Student's t-tests in which we formulate the null hypothesis to be that the distributions are equal. A value of 1 indicates a rejection of the null hypothesis at the 5% significance level. A value of 0 indicates a failure to reject the null hypothesis at the 5% significance level.

Table 4. Experimental Results

Sets	Best model				Sparse model[1]				Baseline	
	dist.	std.dev.	std.err.	feats.	dist.	std.dev.	std.err.	feats.	turns	est. suc.
set 1-2	0.714	0.024	0.008	439.8	0.611	0.025	0.007	11.9	0	0
set 2-3	0.751	0.021	0.007	221.1	0.707	0.032	0.010	14.0	1	1
set 3-4	0.847	0.026	0.008	297.3	0.764	0.032	0.010	13.4	1	1
set 4-5	0.868	0.024	0.008	442.3	0.854	0.017	0.005	12.0	0	0
set 5-6*	0.551	0.021	0.007	16.2	0.565	0.034	0.011	7.5	0	1
set 6-7	0.563	0.031	0.010	175.0	0.570	0.028	0.008	8.9	1	1
set 7-8*	0.523	0.016	0.005	158.9	0.523	0.030	0.009	10.1	0	0
set 8-9	0.952	0.016	0.005	99.2	0.946	0.019	0.006	11.3	1	1

Tables 5-6 report the contribution of each feature in the classification tasks. We report the θ value of equation 1, instead of using a more complicated approach like bootstrapping across different folds as Bach [2008] suggests. Negative values are written in italics for emphasis. Note that for a set of experiments, we only report one of the parameter vectors, since the other one has the same magnitude, but in the opposite direction (sign reversed). In our notation token features start with "T=", and dialogue state features start with "S=". Features that are in the quadratic space appear with a superscript 2. The subscripts represent from what part of the dialogue the feature was extracted, possible values are "win", for features extracted from the beginning window or one of the dialogue states ("greet" for greeting, "conf" for confirmation, etc.). No subscript means that the feature was extracted from the entire dialogue. The "default" feature is the intercept.

Table 5. Weights of features for experiments including sets 1-4

Set 1 - Set 2	θ_1 Set 2 - Set 3	θ_2 Set 3 - Set 4	θ_3 Set 4 - Set 5	θ_4
# of types$_{Greet}$	*-0.09* *Reprompts%*$^2_{Win}$	*-0.17* *S=Ngbhood*$_{Win}$	*-0.20* *Gap user-sys*$_{Conf}$	*-1.30*
NonUnd%$_{Other}$	*-0.08* *# of types*$_{Win}$	*-0.14* NonUnd%	0.19 NonUnd%$^2_{Conf}$	0.13
NonUnd%$^2_{Other}$	*-0.08* S=Ngbhood$_{Win}$	0.10 S=Other	0.18 *S=Other*	*-0.12*
T=Next$_{Win}$	*-0.07* S=Ngbhood	0.04 T=At$_{Win}$	0.12 *Uncovered-Place*	*-0.09*
S=Greet$_{Win}$	*-0.06* *Barge*$^2_{Other}$	*-0.03* *S=Traveltime*	*-0.05* S=Traveltime	0.03

Table 6. Weights of features for experiments including sets 5-8

Set 5 - Set 6	θ_5 Set 6 - Set 7	θ_6 Set 7 - Set 8	θ_7 Set 8 - Set 9	θ_8
T=Hello$_{Win}$	*-0.06* NonUnd%	0.11 *DTMF0*$_{Win}$	*-0.01* Gap user-sys	2.02
NonUnd%	*-0.02* T=D-Zero$_{Win}$	0.04 *Barge*$_{Conf}$	*-0.01* Gap user-sys$_{Conf}$	0.43
Gap sys-user$_{Conf}$	*-0.01* *T=64A*$_{Win}$	*-0.04* *Barge*$^2_{Win}$	*-0.01* <Default>	0.22
T=54C$_{Win}$	0.01 *T=61C*$_{Win}$	*-0.01* *Barge*$^2_{Conf}$	*-0.01* T=Yes$_{Win}$	0.10
Barge$^2_{Win}$	*-0.01* Gap sys-user$_{Conf}$	0.01 *DTMF0*$_{Greet}$	*-0.01* *S=Ngbhood*	*-0.05*

4.4 Discussion

Traditionally, the SDS community evaluates performance in conventional objective metrics, such as mean number of turns and estimated success. Table 3 shows

[1] We report the sparse model using the validation data set, in order to only query once the testing set.

that the standard deviation of the number of turns can be higher than its mean. We notice that these metrics fail to trigger statistically significant differences across the data sets 1-2 and 4-5 (See Table 4). We give evidence that DSDF is more sensitive, since it is able to detect the differences between these sets, and more importantly it is able to describe *what* the differences are. Between sets 1 and 2, the greeting prompt was replaced from a specific prompt to a general "What can I do for you prompt". DSDF is able to detect this change and explain it showing that in the newer system, users' utterances have a richer vocabulary in the greeting dialogue state (see Table 5). We are concerned about the failure of standard metrics to detect changes between sets 4 and 5, since the differences of the back-end infrastructure to Olympus II were a major landmark in the history of the development of Let's Go. DSDF is able to recognize differences in these sets, and explains them as different end pointing strategy and non-understanding percentages. DSDF also provides evidence suggesting that the number of routes supported was incremented, as seen with the different activation of the feature Uncovered-Place$_{Global}$. Again, DSDF is able to accurately describe that the differences between sets 8 and9 are because of different end pointing strategies, agreeing with the development log. As an interesting side note, Table 4 shows that selecting a classifier for its sparsity sometimes has little impact on its performance. The classifier that distinguishes among sets 8 and 9 only gains less than 1% of accuracy by increasing its feature space from 11 to 100.

Estimated success shows a statistically significant difference between sets 5 and 6, even though there was *no* change to the system in these dates. Using DSDF, we explain that differences between these control sets are due to different usage, as the most predictive features are related to differences in the tokens transcribed by the recognizer (see Table 6). More accurately, DSDF considers these control sets as almost indistinguishable, following our initial hypothesis that similar sets should be unrecognizable from each other (55% of distance, out of a minimum of 50%).

DSDF is able to accurately recognize the difference between sets where estimated success and number of turns were unable to detect changes. However it is worth noting how minor the changes between data sets 6 and 7 are considered by DSDF.

5 Conclusions and Future Work

We provided further evidence on the inappropriateness of using conventional metrics to distinguish SDS. We have presented a novel dialogue distance function, based on maximum entropy, that is more fine-grained than conventional metrics and has a lower variance. Furthermore, this model was also used to find unexpected changes in the data sets, and *describe* them.

In future work we will address the issue of drawing a relationship between dialogue differences and an assessment metric. For these we will consider using variable selection to investigate what feature correlate with shorter, more successful dialogues. We will investigate the relationship of the features that help

to distinguish among SDS, and the features that give some information about dialogue quality.

6 Acknowledgements

We thank Antoine Raux and Brian Langner for their guidance on the technical infrastructure. We thank Brian Strope for his suggestions on assessment of SDS. Discussions with Sivaraman Balakrishnan were very helpful.

This project is sponsored by a grant from Google. The first author was partially supported by the Costa Rican Ministry of Science and Technology (MICIT) and the Costa Rican Council for Research in Science and Technology (CONICIT).

References

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

Bacchiani, M., Beaufays, F., Schalkwyk, J., Schuster, M., and Strope, B. (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.

Bach, F. (2008). Bolasso: model consistent Lasso estimation through the bootstrap. In *Proceedings of the 25th international conference on Machine learning*, pages 33–40. ACM New York, NY, USA.

Bishop, C. (1995). *Neural networks for pattern recognition*. Oxford University Press, USA.

Bohus, D., Raux, A., Harris, T., Eskenazi, M., and Rudnicky, A. (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*.

Colmenarez, A. and Huang, T. (1996). Maximum likelihood face detection. In *Int. Conf. Automatic Face and Gesture Recognition*, pages 307–309.

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

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182.

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

Hastie, H. W., Prasad, R., and Walker, M. (2002). Automatic evaluation: Using a date dialogue act tagger for user satisfaction and task completion prediction. In *In LREC 2002*, pages 641–648.

Kamm, C., Walker, M., and Litman, D. (1999). Evaluating spoken language systems. In *Proc. of AVIOS*. Citeseer.

Lee, S., Ganapathi, V., and Koller, D. (2007). Efficient Structure Learning of Markov Networks using L˜ 1-Regularization. *Advances in Neural Information Processing Systems*, 19:817.

McCallum, A. K. (2002). MALLET: A Machine Learning for Language Toolkit. http://mallet.cs.umass.ed.

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

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

Ng, A. Y. (2004). Feature selection, l1 vs. l2 regularization, and rotational invariance. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 78, New York, NY, USA. ACM.

Nigam, K., Lafferty, J., and McCallum, A. (1999). Using maximum entropy for text classification. In *IJCAI-99 workshop on machine learning for information filtering*, pages 61–67.

Paek, T. (2001). Empirical methods for evaluating dialog systems. In *ACL 2001 workshop on evaluation methodologies for language and dialogue systems*, pages 3–10.

Pieraccini, R. (2008). Personal communication.

Raux, A., Bohus, D., Langner, B., Black, A., and Eskenazi, M. (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.

Walker, M., Kamm, C., and Litman, D. (2001). Towards developing general models of usability with PARADISE. *Natural Language Engineering*, 6(3&4):363–377.

Xing, E. P., Jordan, M. I., and Karp, R. M. (2001). Feature selection for high-dimensional genomic microarray data. In *In Proceedings of the Eighteenth International Conference on Machine Learning*, pages 601–608. Morgan Kaufmann.

Zhang, C. and Huang, J. (2008). The sparsity and bias of the lasso selection in high-dimensional linear regression. *Annals of Statistics*, 36(4):1567–1594.