Discriminative state tracking for spoken dialog systems

Angeliki Metallinou[1*], **Dan Bohus**[2], **and Jason D. Williams**[2]
[1]University of Southern California, Los Angeles, CA, USA
[2]Microsoft Research, Redmond, WA, USA
metallin@usc.edu dbohus@microsoft.com jason.williams@microsoft.com

Abstract

In spoken dialog systems, statistical *state tracking* aims to improve robustness to speech recognition errors by tracking a posterior distribution over hidden dialog states. Current approaches based on generative or discriminative models have different but important shortcomings that limit their accuracy. In this paper we discuss these limitations and introduce a new approach for discriminative state tracking that overcomes them by leveraging the problem structure. An offline evaluation with dialog data collected from real users shows improvements in both state tracking accuracy and the quality of the posterior probabilities. Features that encode speech recognition error patterns are particularly helpful, and training requires relatively few dialogs.

1 Introduction

Spoken dialog systems interact with users via natural language to help them achieve a goal. As the interaction progresses, the dialog manager maintains a representation of the state of the dialog in a process called *dialog state tracking*. For example, in a bus schedule information system, the dialog state might indicate the user's desired bus route, origin, and destination. Dialog state tracking is difficult because automatic speech recognition (ASR) and spoken language understanding (SLU) errors are common, and can cause the system to misunderstand the user's needs. At the same time, state tracking is crucial because the system relies on the estimated dialog state to choose actions – for example, which bus schedule information to present to the user.

The dialog state tracking problem can be formalized as follows (Figure 1). Each system turn in the dialog is one datapoint. For each datapoint, the input consists of three items: a set of K features that describes the current dialog context, G dialog state hypotheses, and for each dialog state hypothesis, M features that describe that dialog state hypothesis. The task is to assign a probability distribution over the G dialog state hypotheses, plus a meta-hypothesis which indicates that none of the G hypotheses is correct.

Note that G varies across turns (datapoints) – for example, in the first turn of Figure 1, $G = 3$, and in the second and third turns $G = 5$. Also note that the dialog state tracker is *not* predicting the contents of the dialog state hypotheses; the dialog state hypotheses contents are given by some external process, and the task is to predict a probability distribution over them, where the probability assigned to a hypothesis indicates the probability that it is correct. It is a requirement that the G hypotheses are disjoint; with the special "everything else" meta-hypothesis, exactly one hypothesis is correct by construction. After the dialog state tracker has output its distribution, this distribution is passed to a separate, downstream process that chooses what action to take next (e.g., how to respond to the user).

Dialog state tracking can be seen an analogous to assigning a probability distribution over items on an ASR N-best list given speech input and the recognition output, including the contents of the N-best list. In this task, the general features describe the recognition overall (such as length of utterance), and the hypothesis-specific features describe each N-best entry (such as decoder cost).

[*] Work done while at Microsoft Research

Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pages 466–475,
Sofia, Bulgaria, August 4-9 2013.

Another analogous task is assigning a probability distribution over a set of URLs given a search query and the URLs. Here, general features describe the whole set of results, e.g., number of words in the query, and hypothesis-specific features describe each URL, e.g., the fraction of query words contained in page.

For dialog state tracking, most commercial systems use hand-crafted heuristics, selecting the SLU result with the highest confidence score, and discarding alternatives. In contrast, statistical approaches compute a posterior *distribution* over many *hypotheses* for the dialog state. The key insight is that dialog is a temporal process in which correlations between turns can be harnessed to overcome SLU errors. Statistical state tracking has been shown to improve task completion in end-to-end spoken dialog systems (Bohus and Rudnicky (2006); Young et al. (2010); Thomson and Young (2010)).

Two types of statistical state tracking approaches have been proposed. *Generative* approaches (Horvitz and Paek (1999); Williams and Young (2007); Young et al. (2010); Thomson and Young (2010)) use generative models that capture how the SLU results are generated from hidden dialog states. These models can be used to track an arbitrary number of state hypotheses, but cannot easily incorporate large sets of potentially informative features (e.g. from ASR, SLU, dialog history), resulting in poor probability estimates. As an illustration, in Figure 1, a generative model might fail to assign the highest score to the correct hypothesis (61C) after the second turn. In contrast, *discriminative* approaches use conditional models, trained in a discriminative fashion (Bohus and Rudnicky (2006)) to directly estimate the distribution over a set of state hypotheses based on a large set of informative features. They generally produce more accurate distributions, but in their current form they can only track a handful of state hypotheses. As a result, the correct hypothesis may be discarded: for instance, in Figure 1, a discriminative model might consider only the top 2 SLU results, and thus fail to consider the correct 61C hypothesis at all.

The main contribution of this paper is to develop a new discriminative model for dialog state tracking that can operate over an arbitrary number of hypotheses *and* still compute accurate probability estimates. We also explore the relative importance of different feature sets for this task, and measure the amount of data required to reliably train our model.

2 Data and experimental design

We use data from the public deployment of two systems in the Spoken Dialog Challenge (Black et al. (2010)) which provide bus schedule information for Pittsburgh, USA. The systems, DS1 and DS2, were fielded by AT&T, and are described in Williams et al. (2010) and Williams (2012). Both systems followed a highly directed flow, separately collecting 5 *slots*. All users were asked for their bus route, origin, and destination; then, they were optionally prompted for a date and time. Each slot was explicitly or implicitly confirmed before collecting the next. At the end, bus times were presented. The two systems differed in acoustic models, confidence scoring model, state tracking method and parameters, number of supported routes (8 vs 40, for DS1 and DS2 respectively), presence of minor bugs, and user population. These differences yield distinctions in the distributions in the two corpora (Williams (2012)).

In both systems, a dialog state hypothesis consists of a value of the user's goal for a certain slot: for example, a state hypothesis for the origin slot might be "carnegie mellon university". The number G of state hypotheses (e.g. slot values) observed so far depends on the dialog, and turn within that dialog. For instance, in Fig. 1, G progressively takes values 3, 5 and 5. Dialog state hypotheses with identical contents (e.g., the same bus route) are merged. The correctness of the SLU results was manually labeled by professional annotators.

2.1 Experimental setup

To perform a comparative analysis of various state tracking algorithms, we test them *offline*, i.e., by re-running state tracking against the SLU results from deployment. However, care must be taken: when the improved state-tracker is installed into a dialog system and used to drive action selection, the distribution of the resulting dialog data (which is an input for the state tracker) will change. In other words, it is known *a priori* that the train and test distributions will be *mismatched*. Hence, when conducting offline experiments, if train and test data were drawn from the same *matched* distribution, this may overstate performance.

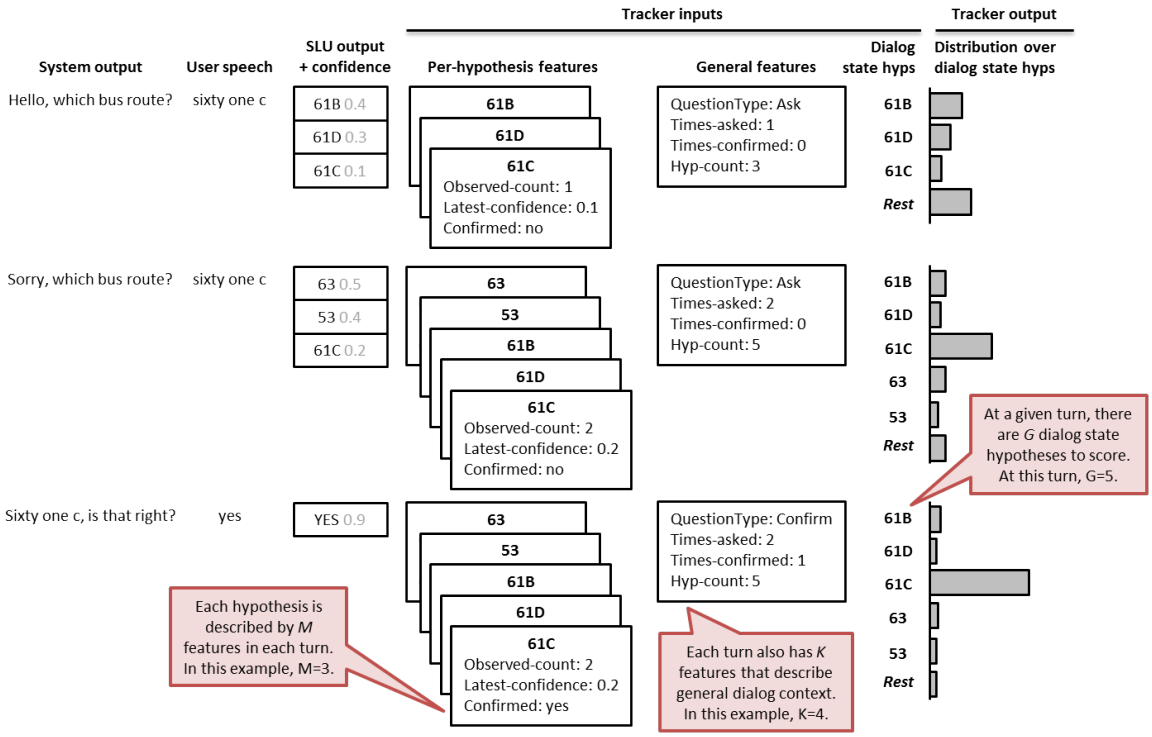


Figure 1: Overview of dialog state tracking. In this example, the dialog state contains the user's desired bus route. At each turn, the system produces a spoken output. The user's spoken response is processed to extract a set of spoken language understanding (SLU) results, each with a *local* confidence score. A set of G dialog state hypotheses is formed by considering all SLU results observed so far, including the current turn and all previous turns. For each state hypothesis, a feature extractor produces a set of M hypothesis-specific features, plus a single set of K general features that describes the current dialog context. The dialog state tracker uses these features to produce a distribution over the G state hypotheses, plus a meta-hypothesis *rest* which accounts for the possibility that none of the G hypotheses are correct.

dataset	train set	test set
MATCH1	half calls from DS2	remaining calls in DS2
MATCH2	half calls from DS1, half from DS2	remaining calls from DS1 and DS2
MISMATCH	all calls from DS1	all calls from DS2

Table 1: Train-test data splits

To account for this effect, we explicitly study train/test mismatch through three partitions of data from DS1 and DS2 (see Table 1): MATCH1 contains matched train/test data from the DS2 dataset; MATCH2 contains matched train/test data from both datasets; finally, MISMATCH contains mismatched train/test data. While the MISMATCH condition may not identically replicate the mismatch observed from deploying a new state tracker *online* (since online characteristics depend on user behavior) training on DS1 and testing on DS2 at least ensures the presence of some real-world mismatch.

We assess performance via two metrics: *accuracy* and *L2 norm*. Accuracy indicates whether the state hypothesis with the highest assigned probability is correct, where *rest* is correct iff none of the SLU results prior to the current turn include the user's goal. High accuracy is important as a dialog system must ultimately commit to a single interpretation of the user's needs – e.g., it must commit to a route in order to provide bus timetable information. In addition, the L2 norm (or Brier score, Murphy (1973)) also captures how well calibrated the output probabilities are, which is crucial to decision theoretic methods for action selection. The L2 norm is computed between the output posterior and the ground-truth vector, which has 1 in the position of the correct item and 0 elsewhere. Both metrics are computed for each slot in each turn, and reported by averaging across all turns and slots.

2.2 Hand-crafted baseline state tracker

As a baseline, we construct a hand-crafted state tracking rule that follows a strategy common in commercial systems: it returns the SLU result with the maximum confidence score, ignoring all other hypotheses. Although this is very a simple rule, it is very often effective. For example, if the user says "no" to an explicit confirmation or "go back" to an implicit confirmation, they are asked the same question again, which gives an opportunity for a higher confidence score. Of the G possible hypotheses for a slot, we denote the number actually assigned a score by a model as $\tilde{G}$, so in this heuristic baseline $\tilde{G} = 1$.

The performance of this baseline (BASELINE in Table 3) is relatively strong because the top SLU result is by far most likely to be correct, and because the confidence score was already trained with slot-specific speech data (Williams and Balakrishnan (2009), Williams (2012)). However, this simple rule can't make use of SLU results on the N-best list, or statistical priors; these limitations motivate the use of statistical state trackers, introduced next.

3 Generative state tracking

Generative state tracking approaches leverage models that describe how SLU results are *generated* from a hidden dialog state, denoted g. The user's true (unobserved) action u is conditioned on g and the system action a via a user action model $P(u|g, a)$, and also on the observed SLU result $\tilde{u}$ via a model of how SLU results are generated $P(\tilde{u}|u)$. Given a prior distribution $b(g)$ and a result $\tilde{u}$, an updated distribution $b'(g)$ can be computed by summing over all hidden user actions u:

$$b'(g) = \eta \sum_u P(\tilde{u}|u) \cdot P(u|g, a)b(g) \quad (1)$$

where η is a normalizing constant (Williams and Young (2007)). Generative approaches model the posterior over *all* possible dialog state hypotheses, including those not observed in the SLU N-best lists. In general this is computationally intractable because the number of states is too large. One approach to scaling up is to group g into a few *partitions*, and to track only states suggested by observed SLU results (Young et al. (2010); Williams (2010); Gašić and Young (2011)). Another approach is to *factor* the components of a dialog state, make assumptions about conditional independence between the components, and apply approximate inference techniques such as loopy belief propagation (Thomson and Young (2010)).

In deployment, DS1 and DS2 used the AT&T Statistical Dialog Toolkit (ASDT) for dialog state tracking (Williams (2010); AT&T Statistical Dialog Toolkit). ASDT implements a generative update of the form of Eq 1, and uses partitions to maintain tractability. Component models were learned from dialog data from a different dialog system. A maximum of $\tilde{G} = 20$ state hypotheses were tracked for each slot. The performance (GENONLINE in Table 3), was worse than BASELINE: an in-depth analysis attributed this to the mismatch between train and test data in the component models, and to the underlying flawed assumption of eq. 1 that observations at different turns are independent conditioned on the dialog state – in practice, confusions made by speech recognition are highly correlated (Williams (2012)).

For all datasets, we re-estimated the models on the train set and re-ran generative tracking with an unlimited number of partitions (i.e., $\tilde{G} = G$); see GENOFFLINE in Table 3. The re-estimated tracker improved accuracy in MATCH conditions, but degraded accuracy in the MISMATCH condition. This can be partly attributed to the difficulty in estimating accurate initial priors $b(g)$ for MISMATCH, where the bus route, origin, and destination slot values in train and test systems differed significantly.

4 Discriminative State Tracking: Preliminaries and existing work

In contrast to generative models, discriminative approaches to dialog state tracking directly predict the correct state hypothesis by leveraging discriminatively trained conditional models of the form $b(g) = P(g|f)$, where f are features extracted from various sources, e.g. ASR, SLU, dialog history, etc. In this work we will use maximum entropy models. We begin by briefly introducing these models in the next subsection. We then describe the features used, and finally review existing discriminative approaches for state tracking which serve as a starting point for the new approach we introduce in Section 5.

4.1 Maximum entropy models

The maximum entropy framework (Berger et al. (1996)) models the conditional probability distribution of the label y given features $\mathbf{x}$, $p(y|\mathbf{x})$ via an exponential model of the form:

$$P(y|\mathbf{x},\lambda) = \frac{\exp(\sum_{i\in I}\lambda_i\phi_i(\mathbf{x},y))}{\sum_{y\in Y}\exp(\sum_{i\in I}\lambda_i\phi_i(\mathbf{x},y))} \quad (2)$$

where $\phi_i(\mathbf{x},y)$ are feature functions jointly defined on features and labels, and λ_i are the model parameters. The training procedure optimizes the parameters λ_i to maximize the likelihood over the data instances subject to regularization penalties. In this work, we optimize the L1 penalty using a cross-validation process on the train set, and we use a fixed L2 penalty based on heuristic based on the dataset size. The same optimization is used for all models.

4.2 Features

Discriminative approaches for state tracking rely on informative features to predict the correct dialog state. In this work we designed a set of *hypothesis-specific* features that convey information about the correctness of a particular state hypothesis, and a set of *general* features that convey information about the correctness of the *rest* meta-hypothesis.

Hypothesis-specific features can be grouped into 3 categories: *base*, *history* and *confusion* features. *Base* features consider information about the *current* turn, including rank of the current SLU result (current hypothesis), the SLU result confidence score(s) in the current N-best list, the difference between the current hypothesis score and the best hypothesis score in the current N-best list, etc. *History* features contain additional useful information about past turns. Those include the number of times an SLU result has been observed before, the number of times an SLU result has been observed before at a specific rank such as rank 1, the sum and average of confidence scores of SLU results across all past recognitions, the number of possible past user negations or confirmations of the current SLU result etc.

Confusion features provide information about likely ASR errors and confusability. Some recognition results are more likely to be incorrect than others – background noise tends to trigger certain results, especially short bus routes like "p". Moreover, similar sounding phrases are more likely to be confused. The confusion features were computed on a subset of the training data. For each SLU result we computed the fraction of the time that the result was correct, and the binomial 95% confidence interval for that estimate. Those two statistics were pre-computed for all SLU results in the training data subset, and were stored in a lookup table. At runtime, when an SLU hypothesis is recognized, its statistics from this lookup table are used as features. Similar statistics were computed for prior probability of an SLU result appearing on an N-best list, and prior probability of SLU result appearance at specific rank positions of an N-best list, prior probability of confusion between pairs of SLU results, and others.

General features provide aggregate information about dialog history and SLU results, and are shared across different SLU results of an N-best list. For example, from the current turn, we use the number of distinct SLU results, the entropy of the confidence scores, the best path score of the word confusion network, etc. We also include features that contain aggregate information about the sequence of all N-best lists up to the current turn, such as the mean and variance of N-best list lengths, the number of distinct SLU results observed so far, the entropy of their corresponding confidence scores, and others.

We denote the number of *hypothesis-specific* features as M, and the number of *general* features as K. K and M are each in the range of $100-200$, although M varies depending on whether *history* and *confusion* features are included. For a given dialog turn with G state hypotheses, there are a total of $G*M+K$ distinct features.

4.3 Fixed-length discriminative state tracking

In past work, Bohus and Rudnicky (2006) introduced discriminative state tracking, casting the problem as standard multiclass classification. In this setup, each turn constitutes one data instance. Since in dialog state tracking the number of state hypotheses varies across turns, Bohus and Rudnicky (2006) chose a *subset* of $\tilde{G}$ state hypotheses to score. In this work we used a similar setup, where we considered the top G_1 SLU results from the current N-best list at turn t, and the top G_2 and G_3 SLU results from the previous N-best lists at turns $t-1$ and $t-2$. The problem can then be formulated as multiclass classification

over $\tilde{G}+1 = G_1+G_2+G_3+1$ classes, where the correct class indicates which of these hypotheses (or *rest*) is correct. We experimented with different values and found that $G_1 = 3$, $G_2 = 2$, and $G_3 = 1$ ($\tilde{G} = 6$) yielded the best performance.

Feature functions are defined in the standard way, with one feature function ϕ and weight λ for each (feature,class) pair. Formally, ϕ of eq. 2 is defined as $\phi_{i,j}(x, y) = x_i\delta(y, j)$, where $\delta(y, j) = 1$ if $y = j$ and 0 otherwise. i indexes over the $\tilde{G}M + K$ features and j over the $\tilde{G} + 1$ classes.[1] The two-dimensional subscript i, j if used for clarity of notation, but is otherwise identical in role to the one-dimension subscript i in Eq 2. Figure 2 illustrates the relationship between hypotheses and weights.

Results are reported as DISCFIXED in Table 3. In the MATCH conditions, performance is generally higher than the other baselines, particularly when confusion features are included. In the MISMATCH condition, performance is worse that the BASELINE.

A strength of this approach is that it enables features from every hypothesis to independently affect every class. However, the total number of feature functions (hence weights to learn) is $(\tilde{G} + 1) \times (\tilde{G}M + K)$, which increases quadratically with the number of hypotheses considered $\tilde{G}$. Although regularization can help avoid overfitting *per se*, it becomes a more challenging task with more features. Learning weights for each (feature,class) pair has the drawback that the effect of hypothesis-specific features such as confidence have to be learned separately for every hypothesis. Also, although we know in advance that posteriors for a dialog state hypothesis are most dependent on the features corresponding to that hypothesis, in this approach the features from *all* hypotheses are pooled together and the model is left to discover these correspondences via learning. Furthermore, items lower down on the SLU N-best list are much less likely to be correct: an item at a very deep position (say 19) might never be correct in the training data – when this occurs, it is unreasonable to expect posteriors to be estimated accurately.

As a result of these issues, in practice $\tilde{G}$ is limited to being a small number – here we found that increasing $\tilde{G} > 6$ *degraded* performance. Yet with $\tilde{G} = 6$, we found that in 10% of turns, the correct state hypothesis was present but was being discarded by the model, which substantially reduces the upper-bound on tracker performance. In the next section, we introduce a novel discriminative state tracking approach that addresses the above limitations, and enables jointly considering an arbitrary number of state hypotheses, by exploiting the structure inherent in the dialog state tracking problem.

[1] Although in practice, maximum entropy model constraints render weights for one class redundant.

5 Dynamic discriminative state tracking

The key idea in the proposed approach is to use feature functions that *link* hypothesis-specific features to their corresponding dialog state hypothesis. This approach makes it straightforward to model relationships such as "higher confidence for an SLU result increases the probability of its corresponding state hypothesis being correct". This formulation also decouples the number of models parameters (i.e. weights to learn) from the number of hypotheses considered, allowing an arbitrary number of dialog states hypotheses to be scored.

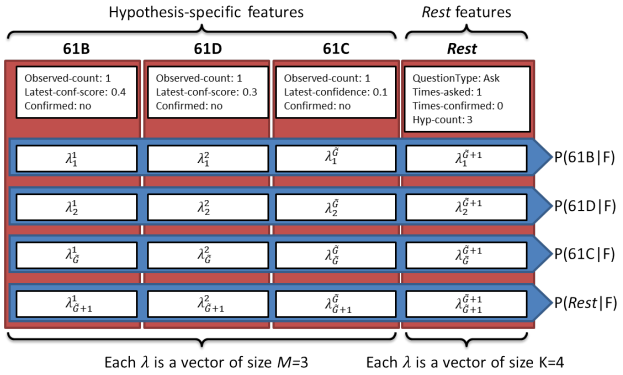


Figure 2: The DISCFIXED model is a traditional maximum entropy model for classification. Every feature in every hypothesis is linked to every hypothesis, requiring $(\tilde{G} + 1)(\tilde{G}M + K)$ weights.

We begin by re-stating how features are indexed. Recall each dialog state hypothesis has M hypothesis-specific features; for each hypothesis, we concatenate these M features with the K general features, which are identical for all hypotheses. For the meta-hypothesis *rest*, we again concatenate $M+K$ features, where the M hypothesis-specific features take special undefined values. We write x_i^g to refer to the ith feature of hypothesis g, where i ranges from 1 to $M + K$ and g from 1 to $G + 1$.

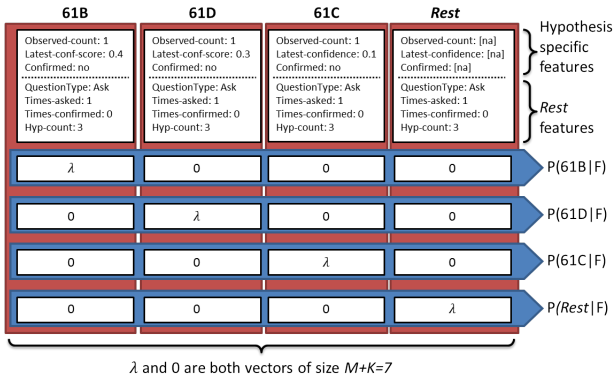


Figure 3: The DISCDYN model presented in this paper exploits the structure of the state tracking problem. Features are linked to only their own hypothesis, and weights are shared across all hypotheses, requiring $M + K$ weights.

algorithm	description
BASELINE	simple hand-crafted rule
GENONLINE	generative update, in deployed system
GENOFFLINE	generative update, re-trained and run offline
DISCFIXED	discr. fixed size multiclass (7 classes)
DISCDYN1	discr. joint dynamic estimation
DISCDYN2	discr. joint dynamic estimation, using indicator encoding of ordinal features
DISCDYN3	discr. joint dynamic estimation, using indicator encoding and ordinal-ordinal conjunctions
DISCIND	discr. separate estimation

Table 2: Description of the various implemented state tracking algorithms

The model is based on $M + K$ feature functions. However, unlike in traditional maximum entropy models such as the fixed-position model above, these features functions are *dynamically defined* when presented with each turn. Specifically, for a turn with G hypotheses, we define $\phi_i(\mathbf{x}, y = g) = x_i^g$, where y ranges over the set of possible dialog states $G + 1$ (and as above $i \in 1 \ldots M + K$). The feature function ϕ_i is dynamic in that the domain of y – i.e., the number of dialog state hypotheses to score – varies from turn to turn. With feature functions defined this way, standard maximum entropy optimization is then applied to learn the corresponding set of $M + K$ weights, denoted λ_i. Fig. 3 shows the relationship of hypotheses and weights.

In practice, this formulation – in which *general* features are duplicated across every dialog state hypothesis – may require some additional feature engineering: for every hypothesis g and *general* feature i, the value of that general feature x_i^g will be multiplied by the same weight λ_i. The result is that any setting of λ_i affects all scores identically, with no net change to the resulting posterior. Nonetheless, *general* features do contain useful information for state tracking; to make use of them, we add conjunctions (combinations) of *general* and *hypothesis-specific* features.

We use 3 different feature variants. In DISCDYN1, we use the original feature set, ignoring the problem described above (so that the *general* features contribute no information), resulting in $M + K$ weights. DISCDYN2 adds indicator encodings of the ordinal-valued *hypothesis-specific* features. For example, rank is encoded as a vector of boolean indicators, where the first indicator is nonzero if $rank = 1$, the second is nonzero if $rank = 2$, and the third if $rank \geq 3$. This provides a more detailed encoding of the ordinal-valued *hypothesis-specific* features, although it still ignores information from the *general* features. This encoding increases the number of weights to learn to about $2(M + K)$.

Finally, DISCDYN3 extends DISCDYN2 by including conjunctions of the ordinal-valued *general* features with ordinal-valued *hypothesis-specific* features. For example, if the 3-way *hypothesis-specific* indicator feature for rank described above were conjoined with a 4-way *general* indicator feature for dialog state, the result would be an indicator of dimension $3 \times 4 = 12$. This expansion results in approximately $10(M + K)$ weights to learn in DISCDYN3.[2]

For comparison, we also estimated a simpler alternative model, called DISCIND. This model consists of 2 binary classifiers: the first one scores each hypothesis in isolation, using the M *hypothesis-specific* features for that hypothesis + the K *general* features for that turn, and outputs a (single) probability that the hypothesis is correct. For this classifier, each hypothesis (not each turn) defines a data instance. The second binary classifier takes the K *general* features, and outputs a probability that the *rest* meta-hypothesis is correct. For this second classifier, each turn defines one data instance. The output of these two models is then calibrated with isotonic regression (Zadrozny and Elkan (2002)) and normalized to generate the posterior over all hypotheses.

[2]We explored adding all possible conjunctions, including real-valued features, but this increased memory and computational requirements dramatically without performance gains.

Metric	Accuracy (larger numbers better)									L2 (smaller numbers better)								
Dataset	MATCH1			MATCH2			MISMATCH			MATCH1			MATCH2			MISMATCH		
Features	*b*	*bc*	*bch*	*b*	*bc*	*bch*	*b*	*bc*	*bch*	*b*	*bc*	*bch*	*b*	*bc*	*bch*	*b*	*bc*	*bch*
BASELINE	61.5	61.5	61.5	63.4	63.4	63.4	**62.5**	62.5	62.5	27.1	27.1	27.1	25.5	25.5	25.5	27.3	27.3	27.3
GENONLINE	54.4	54.4	54.4	55.8	55.8	55.8	54.8	54.8	54.8	34.8	34.8	34.8	32.0	32.0	32.0	34.8	34.8	34.8
GENOFFLINE	57.1	57.1	57.1	60.1	60.1	60.1	51.8	51.8	51.8	37.6	37.6	37.6	33.4	33.4	33.4	42.0	42.0	42.0
DISCFIXED	61.9	66.7	65.3	63.6	69.7	68.8	59.1	61.9	59.3	27.2	23.6	24.4	25.8	21.9	22.4	28.9	27.8	27.8
DISCDYN1	62.0	70.9	71.1	64.4	**72.4**	**72.9**	59.4	61.8	62.3	26.3	**21.3**	**20.9**	25.0	**20.4**	**20.1**	27.7	26.3	25.9
DISCDYN2	62.6	**71.3**	**71.5**	65.7	72.1	72.2	61.9	63.2	**63.1**	26.3	21.4	21.2	24.4	20.5	20.4	**26.9**	**25.8**	**25.4**
DISCDYN3	**63.6**	70.1	70.9	**65.9**	72.1	70.7	60.7	62.1	62.9	**26.2**	21.5	21.4	**24.3**	20.6	20.7	27.1	25.9	26.1
DISCIND	62.4	69.8	70.5	63.4	71.5	71.8	59.9	**63.3**	62.2	26.7	23.3	22.5	25.7	21.8	20.7	28.4	27.3	28.8

Table 3: Performance of the different algorithms on each dataset using three feature combinations. *Base* features are denoted as b, ASR/SLU *confusion* features as c and *history* features as h. Performance for the feature combinations *bh* is omitted for space; it is between *b* and *bc*.

6 Results and discussion

The implemented state tracking methods are summarized in Table 2, and our results are presented in Table 3. These results suggest several conclusions. First, discriminative approaches for state tracking broadly outperform generative methods. Since discriminative methods incorporate many features and are trained directly to optimize performance, this is perhaps unsurprising for the MATCH conditions. It is interesting that discriminative methods are also superior in the more realistic MISMATCH setting, albeit with smaller gains. This result suggests that discriminative methods have good promise when deployed into real systems, where mismatch between training and test distributions is expected.

Second, the dynamic discriminative DISCDYN models also outperformed the fixed-length discriminative methods. This shows the benefit of a model which can score every dialog state hypotheses, rather than a fixed subset. Third, the three variants of the DISCDYN model, which progressively contain more detailed feature encoding and conjunctions, perform similarly. This suggests that a relatively simple encoding is sufficient to achieve good performance, as the feature indicators and conjunctions present in DISCDYN2 and DISCDYN3 give only a small additional increase.

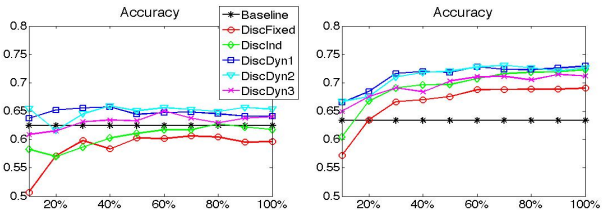
Among the discriminative models, the jointly-optimized DISCDYN versions also slightly outperform the simpler, independently-optimized DISCIND version. This is to be expected, for two reasons: first, DISCIND is trained on a per-hypothesis basis, while the DISCDYN models are trained on a *per-turn* basis, which is the true performance metric. For example, some turns have 1 hypothesis and others have 100, but DISCIND training counts all hypotheses equally. Second, model parameters in DISCIND are trained independently of competing hypotheses. However, they should rather be adjusted specifically so that the correct item receives a larger score than incorrect items – not merely to increase scores for correct items and decrease scores for incorrect items in isolation – and this is what is done in the DISCDYN models.

The analysis of various feature sets indicates that the ASR/SLU error correlation (*confusion*) features yield the largest improvement – c.f. feature set bc compared to b in Table 3. The improvement is smallest for MISMATCH, which underscores the challenges of mismatched train and test conditions during a realistic runtime scenario. Note, however, that we have constructed a highly mismatched case where we train on DS1 (that supports just 8 routes) and test on DS2 (that supports 40 routes). Therefore, many route, origin and destination slot values in the test data do not appear in the training data. Hence, it is unsurprising that the positive effect of confusion features would decrease.

While Table 3 shows performance measures averaged across all turns, Table 4 breaks down performance measures *by slot*, using the full feature set *bch* and the realistic MISMATCH dataset. Results here show a large variation in performance across the different slots. For the date and time slots, there is an order of magnitude less data than for the other slots; however performance for dates is quite good, whereas times is rather poor. We believe this is because the SLU confusion features can be estimated well for slots with small cardinalities (there are 7 possible values for the day), and less well for slots with large cardinalities (there are $24 \times 60 = 1440$ possible time values). This sug-

algorithms	Accuracy (larger numbers better) rout	origin	dest.	date	time
BASELINE	53.81	66.49	67.78	71.88	**52.32**
GENONLINE	50.02	54.11	59.05	75.78	35.02
GENOFFLINE	48.12	58.82	58.98	72.66	20.25
DISCFIXED	52.83	67.81	70.67	71.88	33.34
DISCDYN1	54.28	68.24	68.53	79.69	40.51
DISCDYN2	**56.18**	68.42	70.10	80.47	40.51
DISCDYN3	54.52	66.24	67.96	**82.81**	43.04
DISCIND	54.25	**68.84**	**70.79**	78.13	38.82
algorithms	L2 metric (smaller numbers better) route	origin	dest.	date	time
BASELINE	33.15	24.67	24.68	21.61	**32.35**
GENONLINE	35.50	35.10	31.13	19.86	52.58
GENOFFLINE	46.42	35.73	37.76	19.97	70.30
DISCFIXED	34.09	23.92	23.35	17.59	40.15
DISCDYN1	31.30	23.01	23.07	15.29	37.02
DISCDYN2	**30.53**	**22.40**	**22.74**	**13.58**	37.59
DISCDYN3	31.58	23.86	23.68	13.93	37.52
DISCIND	36.50	23.45	23.41	15.20	45.43

Table 4: Performance per slot on dataset MISMATCH using the full feature set *bch*.



(a) MISMATCH dataset (b) MATCH2 dataset

Figure 4: Accuracy vs. amount of training data

gests that the amount of data required to estimate a good model may depend on the cardinality of slot values.

Finally, in Figure 4 we show how performance varies with different amounts of training data for the MATCH2 and MISMATCH datasets, where the full training set size is approximately 5600 and 4400 turns, respectively. In both cases asymptotic performance is reached after about 2000 turns, or about 150 dialogs. This is particularly encouraging, as it suggests models could be learned or adapted online with relatively little data, or could even be individually tailored to particular users.

7 Conclusion and Future Work

Dialog state tracking is crucial to the successful operation of spoken dialog systems. Recently developed statistical approaches are promising as they fully utilize the dialog history, and can incorporate priors from past usage data. However, existing methodologies are either limited in their accuracy or their coverage, both of which hamper performance.

In this paper, we have introduced a new model for discriminative state tracking. The key idea is to exploit the structure of the problem, in which each dialog state hypothesis has features drawn from the same set. In contrast to past approaches to discriminative state tracking which required a number of parameters quadratic in the number of state hypotheses, our approach uses a constant number of parameters, invariant to the number of state hypotheses. This is a crucial property that enables generalization and dealing with an unlimited number of hypotheses, overcoming a key limitation in previous models.

We evaluated the proposed method and compared it to existing generative and discriminative approaches on a corpus of real-world human-computer dialogs chosen to include a mismatch between training and test, as this will be found in deployments. Results show that the proposed model exceeds both the accuracy and probability quality of all baselines when using the richest feature set, which includes information about common ASR confusions and dialog history. The model can be trained efficiently, i.e. only about 150 training dialogs are necessary.

The next step is to incorporate this approach into a deployed dialog system, and use the estimated posterior over dialog states as input to the action selection process. In future, we also hope to explore unsupervised online adaptation, where the trained model can be updated as test data is processed.

Acknowledgments

We thank Patrick Nguyen for helpful discussions regarding maximum entropy modeling and feature functions for handling structured and dynamic output classification problems.

References

AT&T Statistical Dialog Toolkit. AT&T Statistical Dialog Toolkit. `http://www2.research.att.com/sw/tools/asdt/`, 2013.

Adam Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22:39–71, 1996.

Alan W. Black, S. Burger, B. Langner, G. Parent, and M. Eskenazi. Spoken dialog challenge 2010. In *Proc. of Workshop on Spoken Language Technologies (SLT)*, 2010.

Dan Bohus and Alex Rudnicky. A k hypotheses + other belief updating model. In *Proc. of AAAI Workshop on Statistical and Empirical Approaches to Spoken Dialog Systems*, 2006.

Milica Gašić and Steve Young. Effective handling of dialogue state in the hidden information state pomdp dialogue manager. *ACM Transactions on Speech and Language Processing*, 7, 2011.

Eric Horvitz and Tim Paek. A computational architecture for conversation. In *Proc. of the 7th Intl. Conf. on User Modeling*, 1999.

Allan H Murphy. A new vector partition of the probability score. *Journal of Applied Meteorology*, 12:595–600, 1973.

Blaise Thomson and Steve Young. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech and Language*, 24(4):562–588, 2010.

Jason D. Williams. Incremental partition recombination for efficient tracking of multiple dialogue states. In *Proc. of ICASSP*, 2010.

Jason D. Williams. Challenges and opportunities for state tracking in statistical spoken dialog systems: Results from two public deployments. *IEEE Journal of Selected Topics in Signal Processing, Special Issue on Advances in Spoken Dialogue Systems and Mobile Interface*, 6(8): 959–970, 2012.

Jason D. Williams and Suhrid Balakrishnan. Estimating probability of correctness for asr n-best lists. In *Proc. SigDial Conference*, 2009.

Jason D. Williams and Steve Young. Partially observable markov decision processes for spoken dialog systems. *Computer Speech and Language*, 21:393–422, 2007.

Jason D. Williams, Iker Arizmendi, and Alistair Conkie. Demonstration of AT&T Let's Go: A production-grade statistical spoken dialog system. In *Proc of Workshop on Spoken Language Technologies (SLT)*, 2010.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. The hidden information state model: a practical framework for POMDP-based spoken dialogue management. *Computer Speech and Language*, 24(2):150–174, 2010.

Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proc. of the eighth ACM SIGKDD Intl. Conf on Knowledge Discovery and Data mining*, pages 694–699, 2002.