Multi-domain learning and generalization in dialog state tracking

Jason D. Williams
Microsoft Research, Redmond, WA, USA
jason.williams@microsoft.com

Abstract

Statistical approaches to dialog state tracking synthesize information across multiple turns in the dialog, overcoming some speech recognition errors. When training a dialog state tracker, there is typically only a *small* corpus of well-matched dialog data available. However, often there is a large corpus of *mis-matched* but related data – perhaps pertaining to different semantic concepts, or from a different dialog system. It would be desirable to use this related dialog data to supplement the small corpus of well-matched dialog data. This paper addresses this task as *multi-domain learning*, presenting 3 methods which synthesize data from different slots and different dialog systems. Since deploying a new dialog state tracker often changes the resulting dialogs in ways that are difficult to predict, we study how well each method *generalizes* to unseen distributions of dialog data. Our main result is the finding that a simple method for multi-domain learning substantially improves performance in highly mis-matched conditions.

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 errors in automatic speech recognition (ASR) and spoken language understanding (SLU) 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.

Most commercial systems use hand-crafted rules for state tracking, selecting the SLU result with the highest confidence score observed so far, and discarding alternatives. In contrast, statistical approaches compute a posterior *distribution* over many *hypotheses* for the dialog state, and in general these have been shown to be superior (Horvitz and Paek, 1999; Williams and Young, 2007; Young et al., 2009; Thomson and Young, 2010; Bohus and Rudnicky, 2006; Metallinou et al., 2013).

Unfortunately, when training a dialog state tracker, there is rarely a large corpus of *matched* data available. For example, a pilot version of the system may be fielded in a controlled environment to collect a small initial corpus. Yet there is often a large quantity of *mis-matched* dialog data available. For example, dialog data might be available from another dialog system – such as an earlier version with a different recognizer, dialog controller, and user population – or from a related task – such as searching for restaurants instead of hotels.

In this paper, we tackle the general problem of **how to make use of disparate sources of data** when training a dialog state tracker. For example, should a tracker for each slot be trained on small sets of slot-specific data, or should data from all slots be combined somehow? Can dialog data from another system be used to build effective tracker for a new system for which no data (yet) exists? Once data from the new system is available, is the old data still useful?

These inter-related questions can be formalized as **multi-domain learning** and **generalization**. Multi-domain learning (**MDL**) refers to the task of building a model – here, a state tracker – for

Proceedings of the SIGDIAL 2013 Conference, pages 433–441,
Metz, France, 22-24 August 2013.

a target domain using training data from both the target domain and a different but related domain. Generalization refers to the ability of a model to perform well in a domain unlike that seen in any of the training data. Both multi-domain learning and generalization are active research topics in the machine learning community, with broad applications. (Joshi et al., 2012) provides a comparison of popular methods on several (non-dialog) tasks, including sentiment classification in on-line product reviews.

In dialog state tracking, there are a variety of properties that could be cast as a "domain". In this paper, we explore two obvious domains: different *dialog systems*, and different *slots*, where slots are informational sub-units of the dialog state, such as the origin, bus route, and departure time in a bus timetables spoken dialog system. We apply several methods for MDL across varied dialog systems, slots, and combinations of both. MDL is attractive for dialog state tracking because the distribution across slots and systems is related but *not identical*. For example, the ranges of speech recognition confidence scores for two slots such as bus route and date may be different, or one system may use confirmations much more often than another. Despite these differences, there are useful patterns: regardless of the slot or system, higher confidence scores and responses of "yes" to confirmations provide more certainty. The hope is that MDL can provide a principled way of using all available data to maximize accuracy.

An important problem in dialog state tracking is that deploying a new tracker into production will produce a new distribution of dialog data that may be unlike data observed at training time in ways that are difficult to predict. As a result, it is important to test the *generalization* of dialog state tracking models on data that differs from the training distribution. In this paper, we evaluate each of the MDL approaches on multiple held-out datasets, ranging from well-matched to very mis-matched – i.e., dialog data from the same dialog system, a modified version of the dialog system, and a completely different dialog system.

We show that dialog data from multiple existing systems can be used to build good state trackers for a completely new system, and that a simple form of MDL improves generalization substantially. We also find that, if well-matched data from that new system is available, the effect (positive or negative) of MDL is slight. Since in practice the level of mis-match can be difficult to predict, this suggests that training with (a particular form of) MDL is the safest approach.

This paper is organized as follows. Section 2 describes the algorithm used for state tracking and the dialog data employed. Section 3 then introduces methods for multi-domain learning. Section 4 presents results and Section 5 briefly concludes.

2 Preliminaries

We begin by describing the core model used for dialog state tracking, and the source data. Both of these will be important for the development of the multi-domain learning methods in Section 3.

2.1 Dialog state tracking model

There are two dominant approaches to statistical methods for dialog state tracking. *Generative* approaches use generative models that capture how the SLU results are generated from hidden dialog states (Horvitz and Paek, 1999; Williams and Young, 2007; Young et al., 2009; Thomson and Young, 2010). In contrast, *discriminative* approaches use conditional models, trained in a discriminative fashion to directly estimate the distribution over a set of state hypotheses based on a large set of informative features (Bohus and Rudnicky, 2006). Previous work has found that discriminative approaches yield better performance (Metallinou et al., 2013), so we base our experiments on a discriminative model.

We will assume that each dialog state hypothesis is described by a feature vector $\mathbf{x}$, consisting of $|\mathbf{x}| = X$ features. For example, a feature might be the confidence score of the most recent recognition result corresponding to the hypothesis. Features can also be included which describe the current dialog context, such as how many times the target slot has been requested or confirmed. At a turn in a dialog with index i, there are $N_{(i)}$ dialog state hypotheses, each described by X features. We denote the concatenation of all $N_{(i)}$ feature vectors as $\mathbf{X}_{(i)}$, which has size $XN_{(i)}$.

The dialog state tracking task is to take as input the complete feature vector $\mathbf{X}_{(i)}$, and output a distribution over the $N_{(i)}$ hypotheses, plus an additional meta-hypothesis REST that indicates that none of the hypotheses is correct. For training, labels $y_{(i)}$ indicate which of the $N_{(i)}$ hypotheses is correct, or else if none of them is correct. By con-

Group	Feats/hyp $\lvert\mathbf{X}\rvert$	$\lvert\mathbf{X}^*\rvert$	Corpus	Dialogs	Mismatch to training data
A	90	54	643	TRAIN2	None – same distribution
			715	TEST1	Low
			750	TEST2	Medium
B	90	316	1020	TRAIN3	None – same distribution
			438	TEST3	Low
C	90	0		TEST4	High

Table 1: Corpora used in this paper. $|\mathbf{X}|$ denotes the number of *common* features, and $|\mathbf{X}^*|$ denotes the number of *system-specific* features. The data in systems TEST1 and TEST3 has low mis-match to the training data because they use very similar dialog managers as in TRAIN2 and TRAIN3, respectively. The system in corpus TEST2 used a different dialog manager from TRAIN2, but the same set of system actions, speech recognizer, and TTS, resulting in a medium level of mis-match. The system in corpus TEST4 was completely different from any system in the training data. On average there were approximately 13 system turns and 13 user turns per dialog across all corpora. The TRAIN* corpora are used for training, and the TEST* corpora are used for testing. Complete details of the corpora are given in (Williams et al., 2013).

struction the hypotheses are disjoint; with the addition of the REST 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).

Note that the dialog state tracker is not predicting the *contents* of the dialog state hypotheses: the dialog state hypotheses' contents and features are given by some external process – for example, simply enumerating all SLU values observed so far in the dialog. Rather, the task is to predict a probability distribution over the hypotheses, where the probability assigned to a hypothesis indicates the probability that it is correct.

In our previous work, we developed a discriminatively-trained maximum-entropy model for dialog state tracking (Metallinou et al., 2013). The model estimates a single weight for each feature in $\mathbf{x}$; to keep learning tractable, these weights are shared across all state hypotheses being scored. The model includes L1 and L2 regularization. This model was found to out-perform generative models, rule-based approaches typically used in industry, and competing discriminative approaches. The complete details are given in (Metallinou et al., 2013) and are not crucial to this paper, because the multi-domain learning approaches used here will not modify the learning algorithm, but rather modify the *features*, as described below.

2.2 Dialog data

We use dialog data and evaluation methods from the Dialog State Tracking Challenge (Williams et al., 2013; Williams et al., 2012). This data comes from public deployments of dialog systems which provide bus schedule information for Pittsburgh, USA. Three different research groups – denoted Groups A, B, and C – provided dialog systems. Each group used completely different systems, composed of different speech recognizers, acoustic and language models, language understanding, dialog design, and text-to-speech. The differences between systems from different groups was substantial: for example, Group A and C systems allowed users to provide any information at any time, whereas Group B systems followed a highly directed flow, separately collecting each slot. In addition, Groups A and B fielded several versions of their systems over a multi-year period – these versions differed in various ways, such as acoustic models, confidence scoring model, state tracking method and parameters, number of supported bus routes, presence of minor bugs, and user population. Differences across versions and groups yielded differences in overall performance and distributions in the data (Black et al., 2011; Williams, 2012). Following the dialog state tracking challenge, we use these differences to test the ability of dialog state tracking methods to generalize to new, unseen distributions of dialog data. Table 1 lists the groups, datasets, and the relative

match/mis-match between training and test data.

In this data, there are 9 slots: the bus route, date, time, and three components each for the origin and destination, roughly corresponding to streets, neighborhoods, and points-of-interest like universities. In this paper we will build trackers that operate on slots independently – i.e., at each turn, a total of 9 trackers will each output a ranked list of dialog state hypotheses for its slot.[1] The state hypotheses consist of all of the values for that slot observed so far in the dialog – either in an SLU result or output by the system – plus the meta-hypothesis REST that represents the case that none of the observed values is correct.

Each dialog state hypothesis is described by a set of features extracted from the dialog data. The Dialog State Tracking Challenge provides data from all systems in a standard format, from which we extracted 90 features per dialog state hypothesis. We refer to these as *common features*, because they are available for all systems. We denote the concatenation of all common features for all hypotheses at a given turn as $\mathbf{X}_A$, $\mathbf{X}_B$, or $\mathbf{X}_C$, subscripted based on the system from which they were extracted. In addition, the challenge data includes system-specific information. From the Group A and B logs we extracted 54 and 316 *system-specific* features per hypothesis, respectively. We denote the concatenation of all system-specific features for all hypotheses at a given turn as $\mathbf{X}^*_A$ or $\mathbf{X}^*_B$, subscripted based on the system from which they were extracted. Group C logs provided no additional system-specific information. Examples of features are provided in the Appendix.

3 Multi-domain learning methods

3.1 Models for multi-domain learning

In multi-domain learning (MDL), data instances are of the form $(\mathbf{X}_{(i)}, y_{(i)}, d_{(i)})$, where $\mathbf{X}_{(i)}$ are features for instance i, $y_{(i)}$ is the label for instance i, and $d_{(i)}$ is the *domain* of instance i, where there are a total of D domains. The goal is to build a good model for $P_d(y|\mathbf{X})$ – i.e., to predict the label of an instance given its features *and* domain. A baseline model uses only data from domain d to train $P_d(y|\mathbf{X})$; MDL tackles the problem of how to build models that use data from all domains to improve on this baseline. In this paper, we consider the fully-supervised case, where all of the training data has been labeled.

We explore four ways of constructing models. First, in the **IND** baseline model, we build D separate models using only data from a single domain. Next, in the **POOL** model, the data from all domains is simply pooled together into one large corpus; the single model trained on this corpus is used in all domains. Each feature vector is augmented to include an indicator of the domain $d_{(i)}$ from which it originated, as this has been found to confer much of the benefit of more complex MDL algorithms (Joshi et al., 2012). The POOL model can be viewed as the simplest form of MDL.

Next, the **MDL1** model employs a simple but powerful method for MDL developed by (Daume III, 2007). For each data instance, a *synthetic feature vector* is formed with $D + 1$ blocks of size $|\mathbf{X}|$. Each block is set to all zeros, except for block $d_{(i)}$ and block $D + 1$ which are both set to $\mathbf{X}_{(i)}$. For example, with $D = 3$ domains, the synthetic feature vector for $\mathbf{X}_{(i)}$ from domain 1 would be $\langle \mathbf{X}_{(i)}, \mathbf{0}, \mathbf{0}, \mathbf{X}_{(i)} \rangle$, and for $\mathbf{X}_{(j)}$ from domain 2 would be $\langle \mathbf{0}, \mathbf{X}_{(j)}, \mathbf{0}, \mathbf{X}_{(j)} \rangle$, where $\mathbf{0}$ is a vector of zeros of size $|\mathbf{X}|$. This synthetic corpus is then used to train a single model which is used in any domain.

This approach has been found to be successful on a variety of machine learning tasks, including several NLP tasks (Daume III, 2007). To explain the intuition, consider a single feature component of $\mathbf{X}$, $\mathbf{X}[k]$, which appears $D + 1$ times in the synthetic feature vectors. For model estimation, assume a standard loss function with a term that penalizes classification errors, and a regularization term that penalizes non-zero feature weights. Intuitively, if an individual scalar feature $\mathbf{X}[k]$ behaves *differently* in the domains, the classifier will prefer the per-domain copies, and assign a zero weight to the final copy, reducing the error term of the loss function, at the expense of a small increase in the regularization term. On the other hand, if an individual scalar feature $\mathbf{X}[k]$ behaves *similarly* across domains, the model will prefer to assign a single non-zero weight to the final copy and zeros to the per-domain copies, as this will reduce the regularization term in the loss function. In other words, the classifier will prefer the shared copy when doing so has little impact to accuracy – i.e., the classifier chooses on a feature-by-feature basis when to keep domains separate, and when to pool do-

[1] For simplicity, in this paper we do not consider *joint* state hypotheses, which include more than one slot.

Method	Target Slot	Synthetic feature vector encoding for data from: Slot 1	Slot 2	⋯	Slot 9
SLOTIND	1	$\mathbf{X}_1$	not used	⋯	not used
	2	not used	$\mathbf{X}_2$	⋯	not used
	⋯	⋯	⋯	⋯	⋯
	9	not used	not used	⋯	$\mathbf{X}_9$
SLOTPOOL	all	$\mathbf{X}_1$	$\mathbf{X}_2$	⋯	$\mathbf{X}_3$
SLOTMDL1	all	$\mathbf{X}_1, \mathbf{0}, \ldots, \mathbf{0}, \mathbf{X}_1$	$\mathbf{0}, \mathbf{X}_2, \ldots, \mathbf{0}, \mathbf{X}_2$	⋯	$\mathbf{0}, \mathbf{0}, \ldots, \mathbf{X}_9, \mathbf{X}_9$
SLOTMDL2	1	$\mathbf{X}_1, \mathbf{0}, \mathbf{X}_1$	$\mathbf{0}, \mathbf{X}_2, \mathbf{X}_2$	⋯	$\mathbf{0}, \mathbf{X}_9, \mathbf{X}_9$
	2	$\mathbf{0}, \mathbf{X}_1, \mathbf{X}_1$	$\mathbf{X}_2, \mathbf{0}, \mathbf{X}_2$	⋯	$\mathbf{0}, \mathbf{X}_9, \mathbf{X}_9$
	⋯	⋯	⋯	⋯	⋯
	9	$\mathbf{0}, \mathbf{X}_1, \mathbf{X}_1$	$\mathbf{0}, \mathbf{X}_2, \mathbf{X}_2$	⋯	$\mathbf{X}_9, \mathbf{0}, \mathbf{X}_9$

Table 2: Synthetic features constructed for each multi-domain learning method applied to *slots*. Here, the subscript on $\mathbf{X}$ indicates the *slot* it describes.

mains.

When the number of domains D is large, MDL1 can produce large, sparse synthetic feature vectors, confounding training. **MDL2** addresses this by constructing D separate models; in model d, data from all domains *except* d is pooled into one meta-domain. Then the procedure in MDL1 is followed. For example, for model $d = 1$, instances $\mathbf{X}_{(i)}$ from domain $d_{(i)} = 1$ is represented as $\langle \mathbf{X}_{(i)}, \mathbf{0}, \mathbf{X}_{(i)} \rangle$; data from *all other domains* $d_{(i)} \neq 1$ is represented as $\langle \mathbf{0}, \mathbf{X}_{(i)}, \mathbf{X}_{(i)} \rangle$. This synthetic data is then used to train a model for domain 1.

3.2 Application to dialog state tracking

In this study, we consider two orthogonal dimensions of domain – *systems* and *slots* – and combinations of the two.

Multi-domain learning across *slots* means building a tracker for one slot using dialog data pertaining to that slot, plus data pertaining to other slots. In the experiments below, this is done by treating each of the 9 slots as a domain and applying each of the four MDL methods above. Table 2 specifies the precise form of the synthetic feature vectors for each method.

Multi-domain learning across *systems* means building a tracker for one dialog system using dialog data collected with that system, plus data from other dialog systems. Each of the two corpora in the training data – TRAIN2 from Group A and TRAIN3 from Group B – is treated as a domain. Since only the *common* features are shared across domains (i.e., systems), model complexity can be reduced by building different models depending on the target group – the group the model will be tested on – and including system-specific features only for the target group. For example, when a model will be trained on data from Groups A and B, then tested on data from Group A, we include common features from A and B but system-specific features from only A. Table 3 specifies the precise form of the synthetic feature vectors for each method. Also, when MDL is applied across systems, there are only 2 sources of training data, so MDL2 is identical to MDL1 (and thus isn't shown in the results).

Applying multi-domain learning to both systems and slots is done by composing the two feature synthesis steps. This process is simple but can increase the size of synthetic feature vectors by up to an order of magnitude.

3.3 Evaluation method

In the experiments below, we train dialog state trackers that output a scored list of dialog state hypotheses *for each slot* at each turn in the dialog. For evaluation, we measure the fraction of output lists where the top dialog state hypothesis is correct. A dialog state hypothesis is correct if it corresponds to a slot value which has been recognized correctly. The dialog state tracker may include the meta-hypothesis REST among its hypotheses – this meta-hypothesis is labeled as correct if no correct values have yet been recognized for this slot.

Since most turns contain no information about most slots, we limit evaluation to turns where new information for a slot appears either in the speech recognition output, or in the system output. For

Method	Target group	Synthetic feature vector encoding for data from: Group A	Group B
SYSTEMIND	A	$\mathbf{X}_A, \mathbf{X}_A^*$	not used
	B	not used	$\mathbf{X}_B, \mathbf{X}_B^*$
SYSTEMIND-A	C	$\mathbf{X}_A$	not used
SYSTEMIND-B	C	not used	$\mathbf{X}_B$
SYSTEMPOOL	A	$\mathbf{X}_A, \mathbf{X}_A^*$	$\mathbf{X}_B, \mathbf{0}$
	B	$\mathbf{X}_A, \mathbf{0}$	$\mathbf{X}_B, \mathbf{X}_B^*$
	C	$\mathbf{X}_A$	$\mathbf{X}_B$
SYSTEMMDL	A	$\mathbf{X}_A, \mathbf{X}_A^*, \mathbf{0}, \mathbf{X}_A$	$\mathbf{0}, \mathbf{0}, \mathbf{X}_B, \mathbf{X}_B$
	B	$\mathbf{0}, \mathbf{0}, \mathbf{X}_A, \mathbf{X}_A$	$\mathbf{X}_B, \mathbf{X}_B^*, \mathbf{0}, \mathbf{X}_B$

Table 3: Synthetic features constructed for each multi-domain learning method applied to *systems*. Here, the subscript on $\mathbf{X}$ indicates the *system* it originated from. Asterisk super-scripts indicate system-specific features, which are only included for the group the tracker will be tested on (i.e., the *target group*).

example, in turn i, if a system confirms a bus route, and a date appears in the speech recognition output, both of these slots in turn i will be included when computing average accuracy. If the time slot appears in neither the system output nor anywhere in the speech recognition output of turn i, then the time slot in turn i is excluded when computing average accuracy. The accuracy computation itself was done by the scoring tool from the Dialog State Tracking Challenge, using the *schedule2* accuracy metric for *all* slots (Williams et al., 2013; Williams et al., 2012).

For comparison, we also report performance of a simple rule-based tracker. For each slot, this tracker scans over all values recognized so far in the dialog, and returns the value which has been recognized with the highest *local* SLU confidence score.

4 Results

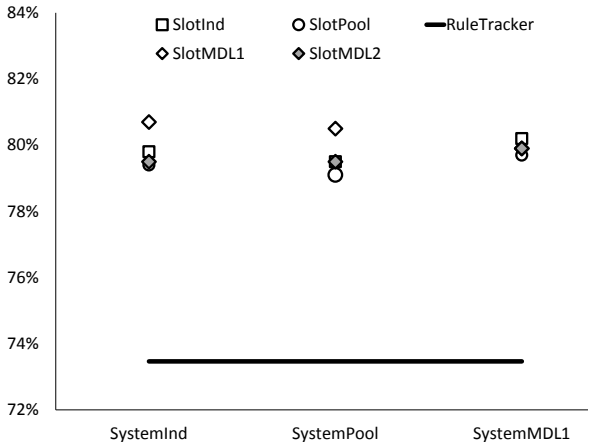
We first evaluated performance of multi-domain learning in isolation, *excluding* the effects of generalization. To do this, we divided TRAIN2 and TRAIN3 in half, using the first halves for training and the second halves for testing. This experiment gives an indication of the performance of multi-domain learning if conditions in deployment match the training data.

Results are shown in Figure 1a-1b. Here, the effects of multi-domain learning across systems and slots is rather small, and inconsistent. For example, pooling slot data yields best performance on TRAIN3, and worst performance in TRAIN2. Applying MDL across systems yields best performance for TRAIN3, but not for TRAIN2. Overall, when training and test data are very well-matched, MDL has little effect.

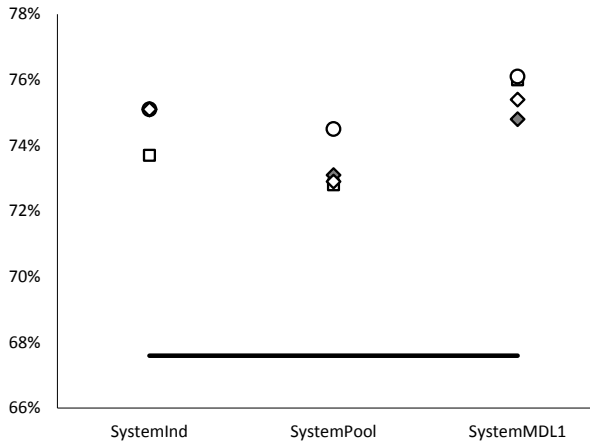
Of course, in practice, training and test data will *not* be well-matched, so we next evaluated performance of multi-domain learning *including* the effects of generalization. Here we trained using the complete TRAIN2 and TRAIN3 corpora, and tested on TEST1, TEST2, TEST3, and TEST4.

Results are shown in Figures 1c-1f. The dominant trend is that, at high levels of mis-match as in TEST3 and TEST4, simply pooling together all available data yields a large increase in accuracy compared to all other methods. The majority of the increase is due to pooling across slots, though pooling across systems yields a small additional gain. This result echos past work, where pooling data is often competitive with more sophisticated methods for multi-domain learning (Joshi et al., 2012).

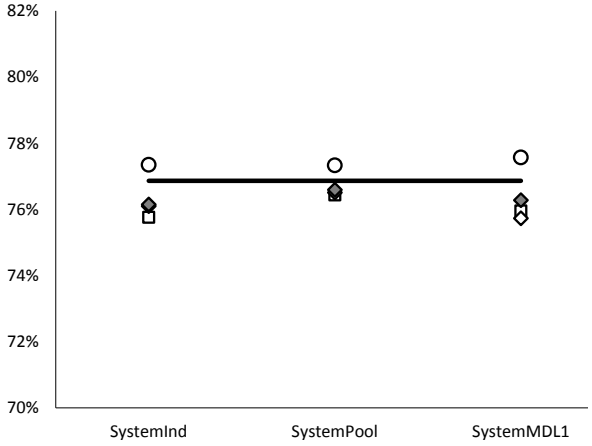
In our case, one possible reason for this result is that simply pooling the data introduces a sort of regularization: note that the models with SLOTPOOL and SYSTEMPOOL have the highest ratio of training data to model parameters. The MDL methods also use all the data, but via their larger synthetic feature vectors, they increase the number of model parameters. The smaller model capacity of the POOL models limit the ability to completely fit the training data. This limitation can be a liability for matched conditions – see for example Figure 1a – but may help the model to generalize



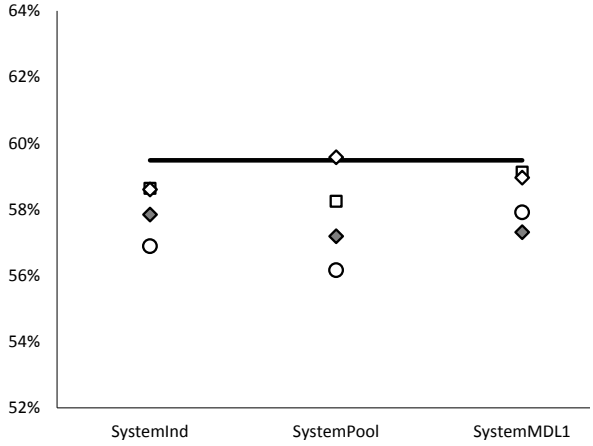
(a) Evaluation on TRAIN2 (Group A), in which there is *minimal* mis-match between the training and test data.



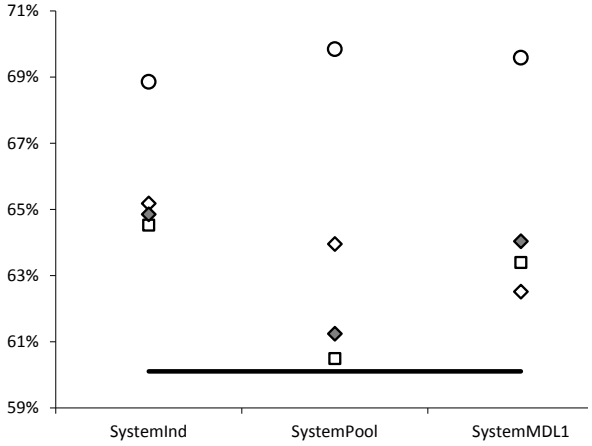
(b) Evaluation on TRAIN3 (Group B), in which there is *minimal* mis-match between the training and test data.



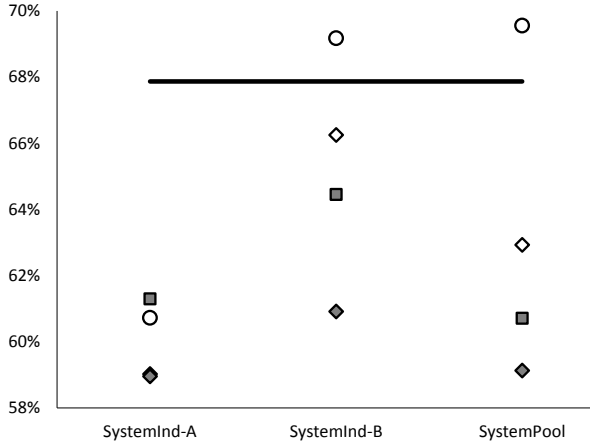
(c) Evaluation on TEST1 (Group A), in which there is *low* mis-match between the training and test data.



(d) Evaluation on TEST3 (Group B), in which there is *low* mis-match between the training and test data.



(e) Evaluation on TEST2 (Group A), in which there is *medium* mis-match between the training and test data.



(f) Evaluation on TEST4 (Group C), in which there is *high* mis-match between all of the training data and test data.

Figure 1: Average accuracy of different approaches to multi-domain learning in dialog state tracking. Squares show SLOTIND, circles SLOTPOOL, unshaded diamonds SLOTMDL1, and shaded diamonds SLOTMDL2. The solid line shows performance of a simple rule-based tracker, which is not trained on data. In all plots, the vertical axis is shown on the same scale for comparability (12% from bottom to top), and indicates average accuracy of the top dialog state (c.f., Section 3.3). In panels 1a and 1b, training is done on the first halves of TRAIN2 and TRAIN3, and testing on the second halves. In the other panels, training uses all of TRAIN2 and TRAIN3. In panel 1f, the categories for TEST4 – for which there is no in-domain data – are different than the other panels.

in mis-matched conditions.

5 Conclusion

This paper has examined multi-domain learning and generalization in dialog state tracking. Two dimensions of domain have been studied – learning across slots and learning across systems – and three simple methods for multi-domain learning have been studied. By using corpora of real dialogs from the Dialog State Tracking Challenge, generalization has been studied through varying levels of mis-match between training and test data.

The results show that simply pooling together data yields large benefits in highly mis-matched conditions and has little effect in well-matched conditions. In practice of course, the level of mismatch a new tracker will produce is difficult to predict. So the safest strategy seems to be to always pool together all available data.

There are a variety of issues to examine in future work. First, the MDL methods used in this study were chosen for their simplicity and versatility: by augmenting features, no changes were required to the learning method. There exist other methods of MDL which *do* modify the learning, and in some cases yield better performance. It would be interesting to test them next, perhaps including methods that can construct deeper representations than the maximum entropy model used here.

More broadly, this study has been limited to *supervised* multi-domain learning, in which *labeled* data from multiple domains is available at training time. It would clearly be desirable to develop a method for *unsupervised* adaptation, in which the model is adjusted as the *unlabeled* test data is experienced.

For now, the contribution of this study is to provide at least an initial recommendation to practitioners on how to best make use of disparate sources of dialog data when building a statistical dialog state tracker.

Acknowledgements

Thanks to Dan Bohus for making his machine learning software available.

References

Alan W Black, Susanne Burger, Alistair Conkie, Helen Hastie, Simon Keizer, Oliver Lemon, Nicolas Merigaud, Gabriel Parent, Gabriel Schubiner, Blaise Thomson, Jason D. Williams, Kai Yu, Steve Young, and Maxine Eskenazi. 2011. Spoken dialog challenge 2010: Comparison of live and control test results. In *Proc SIGdial Workshop on Discourse and Dialogue, Portland, Oregon*.

Dan Bohus and Alex Rudnicky. 2006. A 'K hypotheses + other' belief updating model. In *Proc American Association for Artificial Intelligence (AAAI) Workshop on Statistical and Empirical Approaches for Spoken Dialogue Systems, Boston*.

Hal Daume III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic, June. Association for Computational Linguistics.

Eric Horvitz and Tim Paek. 1999. A computational architecture for conversation. In *Proc 7th International Conference on User Modeling (UM), Banff, Canada*, pages 201–210.

Mahesh Joshi, Mark Dredze, William W Cohen, and Carolyn Rose. 2012. Multi-domain learning: When do domains matter? In *Proc Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Jeju, Korea*.

Angeliki Metallinou, Dan Bohus, and Jason D. Williams. 2013. Discriminative state tracking for spoken dialog systems. In *Proc Association for Computational Linguistics, Sofia*.

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

Jason D Williams and Steve Young. 2007. Partially observable Markov decision processes for spoken dialog systems. *Computer Speech and Language*, 21(2):393–422.

Jason D Williams, Antoine Raux, Deepak Ramachandran, and Alan W Black. 2012. Dialog state tracking challenge handbook. Technical report, Microsoft Research.

Jason D. Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Submitted to SigDial 2013*.

Jason D. Williams. 2012. 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.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2009. The hidden information state model: a practical framework for POMDP-based spoken dialogue management. *Computer Speech and Language*, 24(2):150–174.

Appendix

Example *common features* extracted for all systems
Number of times slot value has been observed in any previous speech recognition result
Whether the most recent speech recognition result includes this slot value
The highest rank on the speech recognition N-best list that this slot value has been observed
The number of times this slot has been requested by the system
Whether the system requested this slot in the current turn
The number of items on the current speech recognition N-best list
Whether confirmation for this slot has been attempted
If confirmation for this slot has been attempted, whether the user was recognized as saying "yes"
The fraction of recognitions of this slot value in the training set which were correct
The fraction of dialogs in the training set in which the user requested this slot value
Example *system-specific features* extracted for Group A systems
Acoustic model score
Average word confidence score
Whether barge-in was triggered
Decoder score
Language model score
Maximum and minimum confidence score of any word
Estimated speaking rate
Estimated speaker gender (male/female)
Example *system-specific features* extracted for Group B systems
Score of best path through the word confusion network
Lowest score of any word on the best path through the word confusion network
Number of speech frames found
Decoder cost
Garbage model likelihood
Noise model likelihood
Average difference in decoder cost, per frame, between the best path and any path through the lattice
Whether barge-in was triggered

Table 4: Examples of features used for dialog state tracking. Group C logs provided no system-specific information.