Learning Models for Speaker, Addressee and Overlap Detection from Multimodal Streams

Oriol Vinyals
University of California at Berkeley
1947 Center Street, Suite 600
Berkeley, CA, US
vinyals@eecs.berkeley.edu

Dan Bohus
Microsoft Research
One Microsoft Way
Redmond, WA, US
dbohus@microsoft.com

Rich Caruana
Microsoft Research
One Microsoft Way
Redmond, WA, US
rcaruana@microsoft.com

ABSTRACT
A key challenge in developing conversational systems is fusing streams of information provided by different sensors to make inferences about the behaviors and goals of people. Such systems can leverage visual and audio information collected through cameras and microphone arrays, including the location of various people, their focus of attention, body pose, the sound source direction, prosody, and speech recognition results. In this paper, we explore discriminative learning techniques for making accurate inferences on the problems of speaker, addressee and overlap detection in multiparty human-computer dialog. The focus is on finding ways to leverage within- and across-signal temporal patterns and to automatically construct representations from the raw streams that are informative for the inference problem. We present a novel extension to traditional decision trees which allows them to incorporate and model temporal signals. We contrast these methods with more traditional approaches where a human expert manually engineers relevant temporal features. The proposed approach performs well even with relatively small amounts of training data, which is of practical importance as designing features that are task dependent is time consuming and not always possible.

Categories and Subject Descriptors
H.1.2 **[Models and Principles]:** User/Machine System – *Human Information Processing*; H.5.2 [**Information Interfaces and Presentation**]: Multimedia Information Systems – *Audio input/outputs;* User Interfaces – *Natural Language*

General Terms
Algorithms, Human Factors

Keywords
Multiparty turn taking; multimodal systems; multimodal inference; addressee detection; overlap detection; speaker identification; learning with multimodal temporally streaming data; random forests

1. INTRODUCTION
One of the key challenges in multimodal interactive systems is performing accurate inference from multiple streams of evidence. Examples include: recognizing activities from joint location information produced by a skeletal tracker, classifying pen or multi-touch gestures, detecting affect from temporal patterns in the acoustic prosodical contour, fusing information from the trajectory and temporal dynamics of visual focus of attention to recognize engagement actions, etc.

A wide array of machine learning techniques have been explored for this type of multimodal inference problems. A simple, commonly used approach for leveraging temporally streaming evidence in classification problems is for a domain expert to manually design features that capture or encode patterns in the temporal streams that may be relevant for the task at hand. These hand-coded features can then be used in conjunction with various machine learning techniques to construct a classification model.

Consider the problem of detecting whether an utterance is addressed to a conversational system or to another participant in a multiparty spoken dialog setting. Apart from basic features that may describe various aspects of the utterance, such as the current dialog state, whether the utterance was correctly understood by the system, the duration of the utterance, etc., a number of *temporally streaming features*, such as the energy in the acoustic signal, its prosodical contour, the frame-by-frame visual focus of attention of the people in the conversation, etc., may also be available. To use the information in these streams with traditional classification approaches, derived features can be manually constructed. Examples include: the average energy at the beginning, middle, and end of the utterance, the percentage of time the speaker maintained their focus-of-attention on the system in the last 100, 200, 500, 1000 milliseconds, prosodical contour statistics like pitch slope, mean, variance, in different segments of the utterance, etc.

This manual feature construction process provides an opportunity for a human expert to infuse deep knowledge into the model by creating complex features that are likely to be relevant to the prediction task. Relying on a manual feature engineering process, however, has several drawbacks: it requires human insight that might not be available for all streams, it makes it more difficult to add additional streams to an existing system, and the expert can never be certain that they have thoroughly explored the space of constructed features.

In this paper we explore the use of discriminative learning techniques on four multimodal inference problems related to speaker, addressee and overlap detection in multiparty human-computer interaction. Developing accurate models for these tasks is critical for enabling physically situated spoken language interaction in open-world settings. We introduce machine learning techniques that aim to shortcut the manual feature engineering

ICMI '12, October 22–26, 2012, Santa Monica, California, USA.

process by directly using the raw streaming features. Specifically, we extend the traditional decision tree and random forest algorithms in a manner that allows them to directly incorporate temporally streaming evidence. We compare this novel methodology to the more common manual feature engineering approach. The results suggest that the new approach can learn directly from the stream features. While the approach does not achieve the full performance of models trained on human engineered features, it eliminates the challenging process of manually constructing task-dependent features.

We begin by reviewing related work in the next section. Then, in Section 3, we describe the technical details of the proposed approach. In Section 4 we describe the datasets and multimodal inference problems we used as the empirical test-bed for the proposed techniques, and in Section 5 we present and discuss in detail the results we obtained. Finally, in Section 6 we draw conclusions and present directions for future work.

2. RELATED WORK

The work described in this paper is part of a larger research effort aimed at developing conversational agents that can engage in spoken language interaction in multiparty settings [1, 2]. Specifically, we investigate a number of discriminative learning techniques on the problems of speaker, addressee and overlap detection in multiparty human-computer interactions. The problems of constructing models for addressee and source identification in multiparty settings, as well as other related problems (*e.g.* tracking visual focus of attention) have been previously investigated in various human-computer, *i.a.* [3, 4] and human-human, *i.a.* [5, 6, 7] settings. For instance in [3] the authors interpolate models that use audio and visual cues to determine addressee in a human-human-robot interaction setting. In [4] the authors explore features such as eye gaze, dialogue history and utterance length, and combine them with a Naïve Bayes classifier to identify addressees. In [5] Bayesian Network and Naïve Bayes classifiers are used for addressee identification in face-to-face meetings with four participants. [6] presents a hierarchical generative approach that is used in conjunction with MCMC techniques for jointly tracking the location and speaking activity of multiple participants in a sensor equipped meeting room. In [7] the authors use a generative model for inferring conversational regime (*i.e.* a characterization of the joint gaze patterns) and head directions in a multiparty conversation. Related work in speaker diarization aiming at very short interactions [8] has shown that for extremely short utterances like the ones present in our scenario classic diarization approaches are prone to failure.

We focus on using discriminative techniques to learn from multiparty human-computer interaction data. Some of the main challenges in our domain include integrating temporally streaming features (*e.g.* information from a microphone array or face tracker) with point features (*e.g.* information about dialog state, etc.); the availability of relatively small amounts of training data; and noisy observations.

Almost all applied machine learning uses domain knowledge to extract meaningful features. A good example of this with temporal sequences and Naïve Bayes decision rules can be found in [9]. In [10] the authors present an approach where a manually constructed dictionary of encoding templates is used to create features over sequences that are then filtered in a feature selection process and integrated with a LDCRF model to recognize head gestures.

The use of decision trees in conjunction with sequential data has been previously investigated in other domains. In [11], the authors used acoustic data to predict phonemes as the acoustic model for speech recognition. In their work, each data point in a sequence is treated as an observation, which is equivalent to stacking all the sequences into a large feature vector, and learning a standard decision tree.

[12] also proposed to learn features from sequences of data, and extended the set of classical rules for building a decision tree to include time-aware splitting criterion. This work is more closely related to one of our approaches, presented in Section 3.1.2, but would be computationally prohibitive given the length and nature of our sequences (which mostly consist of continuous real data instead of discrete observations).

3. APPROACH

We focus our attention on two discriminative learning approaches: maximum entropy models and random forests of decision trees. Maximum entropy models are a robust machine learning technique: they are simple, linear models, are easy to train and produce well-calibrated probability outputs. Decision trees can construct more complex decision surfaces by shaping and combining features in non-linear ways. This property can be important for multimodal inference where relationships between information collected across multiple channels is often relevant for the task at hand. A random forest of decision trees [13] typically is resistant to overfitting and often has good generalization error.

A common approach for learning from streaming evidence with these discriminative learning methods is to manually engineer derived features. In an effort to investigate whether we can shortcut this manual feature engineering process, we propose an extension to decision trees that enables them to handle raw, temporally streaming features. Below, we describe this extension, and discuss in more detail the benefits of the random forest approach.

3.1 Stream-enhanced decision trees

Decision trees construct complex decision surfaces by greedily splitting the data according to one feature (dimension) at a time, in a manner that separates data into neighborhoods of increasing purity. The construction of a decision tree is done in a recursive manner. Given a set of labeled data-points, a search is performed to identify the feature, and a corresponding threshold on that feature that "best" splits or partitions the data into subsets. "Best" is typically defined by various entropy-like metrics, such as information gain, that measure the relative purity of the newly formed subsets in contrast to the original set. Once the best split has been identified, the algorithm is recursively applied on the resulting partitions. The sequence of splits on each path from the root of the tree to a leaf allows trees to learn non-linear combinations of features.

Standard decision trees can easily handle real-valued, ordinal, nominal, and Boolean features. In this work we seek to additionally enable decision trees to reason directly about temporally streaming features. Let us denote a *temporally streaming feature* by $f_{\rightarrow}^{s} = \langle f_1^s, f_2^s, \dots, f_n^s \rangle$, where n is a constant[1], and $f_j^s \in \mathbb{R}$. For instance, if $f_{\rightarrow}^{s}$ is the horizontal location of a face in an image, $f_1^s, f_2^s, \dots, f_n^s$ represent the successive real-values of

[1] Here we assume fixed length streams, but this can be generalized to streams of arbitrary length.

the face location at times 1 through n. The challenge then becomes defining a way to partition data based on a temporally streaming feature $f_{\rightarrow}^{s}$. If an informative partitioning can be done, the traditional decision tree methodology can be applied to leverage the raw streaming features. Below, we discuss two mechanisms for creating such partitions.

3.1.1 The Pivot Approach

The first approach we propose is based on selecting a *pivot* data-point, and creating partitions based on the similarity of other data-points to this pivot, where similarity is computed on the given streaming feature $f_{\rightarrow}^{s}$ using a certain metric.

In traditional decision trees, a split on a non-streaming, real-valued point feature f^{b} is defined by a threshold t. Given the threshold, the data is partitioned into two subsets: one that contains the data-points where $f^{b} < t$ and the other that contains the data-points where $f^{b} \geq t$. In contrast, in the proposed *pivot approach*, a split on a temporally streaming feature $f_{\rightarrow}^{s}$ is defined by a pivot data-point p and a threshold t. Given a pivot and a threshold, the data-points are separated into those that are more similar to the pivot (*i.e.* their distance to the pivot is below t), and those less similar to the pivot (*i.e.* their distance to the pivot is above t) – see Figure 1. The pivots are selected from among the data-points at the split, and thresholds are applied to distance metrics that measure how similar the feature stream $f_{\rightarrow}^{s}$ for one point (the pivot) is to another point that will pass into the left or right branch of the test. For example, if the similarity metric is cosine distance, the test measures how similar each data point is to the corresponding stream of the pivot (using cosine distance): the more similar points go down the left branch of the tree and the less similar points go down the right branch. The separation surface is a sphere in distance metric space, centered on the pivot point. For a real-valued point feature f^{b} the search for the best split involves searching over all possible thresholds, but in the case of a stream feature $f_{\rightarrow}^{s}$, the search must be done over all possible pivots *and* thresholds.

In the approach described so far, similarity is computed on the entire stream, from 1 to n. In reality only certain sub-regions of the stream might be relevant for a given classification problem, so a simple extension of the proposed approach is to compute the similarity on a particular window in the stream $w_s{:}w_e$. In this case, search involves finding a pivot p, a window $w = w_s{:}w_e$, and a threshold t such that the info-gain of the resulting partitions is maximized. The search can be further extended to include multiple distance metrics.

The pivot approach requires that at each node we save not only the selected feature and threshold but also the data-point used to define the pivot so that the similarity of future test points to the pivots can be measured. In traditional decision trees all training data is discarded once the tree has been trained. One can view pivot-based splits as a form of memory-based learning where distance is measured on a stream feature. A sequence of pivot tests performed on different streams in a path in the decision tree can capture complex within- and cross-stream temporal relationships. One potential disadvantage of pivot-based splits is that the number of possible pivot and threshold combinations can be very large. If multiple windows and distance metrics are used to further refine the similarity measure, the search space grows even larger. While the potential size of the search space is very large, by using a random forest decision tree growing procedure we are able to explore only a fraction of the full search space

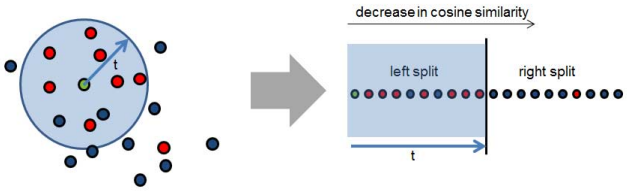

Figure 1. Each point represents a temporal sequence in a high dimensional space. Red and blue are the two classes. The green point is the selected pivot. A metric such as cosine distance is used to measure similarity to the pivot and a radius of threshold t defines the split.

while simultaneously reducing the overfitting of individual trees and increasing the diversity of the tree forest, thus yielding a stream-based decision tree algorithm that is both accurate and reasonably efficient. It is not as efficient as if a human expert had carefully constructed exactly those stream features needed for the task at hand, but it is efficient enough to run on real data and allows one to develop a complete system much faster than when features must be manually constructed.

3.1.2 The Basis-Function Approach

The second approach we investigate does not require memorization of pivot data-points, and is more similar to the traditional feature construction methodology. In order to support the construction of a split on a stream feature $f_{\rightarrow}^{s}$, we compute a set of basis functions within a specific window on the stream, and threshold the result of that function. For example, to create a split on the sound amplitude streaming feature, we might compute the maximum value of the amplitude within the last 200ms in the stream, and search for a threshold on this maximum value. All data points with maximum amplitude in that window below the threshold go down one branch of the tree, while all above the threshold go down the other branch.

Our goal is to develop a small set of basis functions that would be useful across a wide range of problems and streams. We initially used the min, max, and first, second and third moments of the stream. Later we added the second and third *central* moments of the stream (see discussion in section 5).

Unlike the pivot approach, the basis-function approach is equivalent to constructing derived features for each basis function and window in the stream. These features can therefore be pre-computed and then used as regular point features in the decision tree; no extra memorization is required as with pivots. Unlike manually constructed features, once an adequate set of basis functions for temporally streaming features have been established, no manual expertise or insight into the problem or the specific streams is necessary. This makes adding new streams and working with a large number of streams easier.

3.2 Random Forests

A random forest [13] is an unweighted ensemble of decision trees. Each decision tree in the forest is grown on a random subsample of the training data (as with bagging [14]) and uses a randomized split selection procedure that further increases the diversity among the trees. In traditional decision trees, the split selection procedure examines all possible splits and selects the feature and threshold combination that maximizes a measure such as information gain. In random forests, the split selection procedure is only allowed to select from a small random subset of the available features. This randomization makes the decision tree growing process less greedy, increases the diversity of the trees in the ensemble, and

reduces overfitting. Importantly, the random selection of a subset of features from which to select the best split is repeated as each test is installed in the tree, insuring that important attributes and attribute combinations eventually can be discovered on most paths. Although randomization does slightly hurt the accuracy of individual trees, on most problems the gain from the increased diversity of the trees in the ensemble usually more than compensates for this loss, and random forests usually are more accurate than bagged decision trees which do not perform feature randomization prior to selection [15].

In addition to making the ensemble more accurate, a secondary benefit of feature randomization is that far fewer tests need to be considered when deciding what split to install at a node. For example, if only 10% of the available features will be considered for each test, learning will be sped up by a factor of almost 10X. (Because the trees in a random forest are more diverse, one often needs to add more trees to the random forest ensemble, so in practice the gains are somewhat less than this calculation suggests.) We exploit this feature of random forests to partially mitigate the increase in computation required to train trees with the pivot and basis-function approaches outlined above, and to help prevent overfitting from the larger search space over splits.

One advantage of user-constructed features is that complex operations on multiple streams can be crafted prior to learning and are thus available for use in a single split. Tree-based learning methods such as random forests use sequences of tests to combine features. If user-constructed features are not available and the tree must combine multiple streams, the greedy tree-growing process will require multiple steps of learning to construct these higher-order features, and may miss useful combinations if the streams that must be combined do not appear useful individually. Despite this limitation, tree-based ensemble methods are remarkably accurate in many domains [15].

4. DATA

The data for our experiments were collected via a user study that was part of a larger research effort [1, 2] aimed at developing and evaluating computational models for multiparty turn taking. In this experiment, a virtually embodied conversational agent (equipped with a wide-angle camera and a microphone array) engaged in spoken dialog and hosted a trivia questions game with groups of two or three people. The agent asked the questions and mediated the interaction between the participants. The participants could talk to the agent as well as to each other throughout the interactions. Once agreement between participants had been established, the agent indicated if the answer was correct, or provided the correct answer and a short explanation, and then moved to the next question.

The data collection involved 15 groups of four participants each, recruited in pairs of people that knew each other. For each group of four people, all possible subgroups of 2 and of 3 people were formed, and each subgroup played one game with the system. This resulted in a total of 150 dialogs, where in 90 of them the agent played the game with two participants, and in 60 of them the agent played the game with three participants.

The data was transcribed and manually labeled with speaker and addressee information, and an evaluation of the system's turn-taking capabilities was conducted [2]. Lessons learned from this evaluation indicated that major challenges for multiparty turn-taking in these settings include diarization, as well as identifying the speaker and addressees for each utterance. Based on this analysis, we formulated four multimodal inference problems, which we describe below.

4.1 Multimodal Inference Problems

A data-point in each of the problems described below consists of a *detected utterance, i.e.* a continuous segment identified at runtime by the voice activity detector in the system as a single utterance. Given the challenges of diarization in a live, multiparty, interactive setting, each detected utterance may contain speech from a single participant, or concatenated or overlapping speech from multiple participants, or just background noise.

The first problem we defined, which we refer to as the *Target* problem, is a binary classification problem: the goal is to identify whether a detected utterance was addressed to the system or to another participant. When detected utterances contained speech from multiple participants, the addressee for the last spoken segment was considered the target.

The second problem, which we call the *Overlap* problem, is also a binary classification problem: the goal here is to identify whether a detected utterance contains speech from a single, or from multiple participants; the later includes overlapped speech or successive speech from multiple speakers that has been concatenated by the voice-activity detector into a single detected utterance. Detected utterances containing background noise are labeled as single source in the *Overlap* problem.

The third and fourth problems are multiclass problems, and they regard speaker identification. *Source-2* is a 4-way classification problem that was defined on the subset of the data where the system interacted with two participants. The challenge here is to identify whether the detected utterance contains: (1) speech from the first participant, (2) speech from the second participant, (3) speech from both participants, or (4) background noise. Similarly, *Source-3* was defined as a 5-way classification problem on the subset of the data where the system interacted with three participants. The challenge is to identify whether the detected utterance contains (1) speech from the first participant, (2) the second participant, (3) the third participant, (4) a combination of speech from multiple participants, or (5) background noise. The last two problems are closely related to the *Overlap* problem: they refine the *Overlap* detection problem by further separating out the background class and decomposing the "single-source" class to identify the speaker.

4.2 Features

We extracted a set of basic features from seven different modalities and knowledge sources in the system, which we describe below. Some of these features are inherently defined on a per-utterance basis: we refer to them as *basic point features*. Examples include the dialog state at the time the utterance was produced, the speech recognition confidence score for the utterance, whether a non-understanding was triggered, etc. Other features are continuously streaming through time, *e.g.* the microphone array beam direction, or the position of a participant's detected face. We refer to this latter category as *stream features*.

As previously discussed, a common approach for leveraging such streaming evidence is to manually engineer derived point features from the stream features. For instance, computing the standard deviation of the microphone beam location while the utterance is being produced could be informative for predicting whether the detected utterance contains speech from a single source, or overlapping speech from multiple participants. We manually engineered a wide array of such derived features. A large number of these features were designed and used in previous work [16].

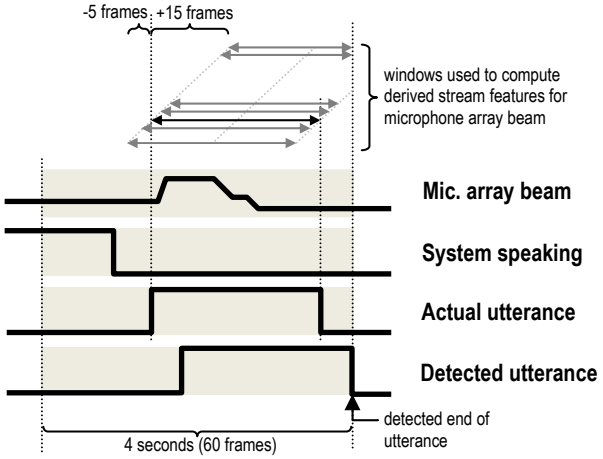

Figure 2. Visualization of four streaming features and windows used to construct derived features for the location of the microphone array beam

To facilitate comparison, both the stream-enabled trees and the manual feature construction were based on the 4-second interval (60 frames at 15fps) prior to the moment the system detected the end of that utterance – see Figure 2. As described below, various sub-regions of this 4-second interval were used to construct derived features.

We now review the set of basic and manually derived features used in the experiments below. They fall into seven classes, according to the knowledge source or modality that originated the feature.

Understanding. This class includes 13 basic point features describing the results of the spoken language recognition and understanding process on the detected utterance. Examples include the speech recognition confidence score, whether or not a non-understanding was triggered, the number of alternates in the n-best list, and average number of words, as well as indicators about whether the top hypothesis or the alternates contain the positive and negative lexical markers *yes* or *no*.

Situation. This class includes one basic point feature indicating whether the agent is interacting with two or three participants.

Dialog. This class includes 8 basic point features that characterize the current dialog state, including classifications at different granularities of the dialog act, whether the last system question was addressed to a single or to multiple participants, whether the previous agent utterance was interrupted by the participant, etc.

Turn-taking. This class includes 3 binary stream features, indicating: (1) when the detected utterance is in progress, (2) when the actual utterance is in progress, and (3) when an agent utterance is in progress. Because of technological delays in the voice activity detection and speech processing pipeline, there is a delay between the moment an utterance actually begins (or ends) until it is detected by the speech engine – see also Figure 2. Based on information available in the 4-second window on these three stream features, we constructed an additional 16 derived features that describe various temporal aspects of turn taking, such as the duration of the detected and actual utterances, and of system speech, the deltas between system speech and the utterances, and the presence of overlaps.

Microphone array. This class includes 3 stream features, which indicate (at two different granularities) the location of the active microphone array beam at any point in time. To construct derived features we considered a set of windows of the same size as the actual utterance but shifted with respect to the actual utterance by -5, -4, -3, … , +14, +15 frames – see Figure 2. If a shifted window extended beyond the boundaries of the 4 second segment, it was cropped to those boundaries. Within the defined windows, we computed the number of distinct active beams, the beam excursion and standard deviation, as well as the average absolute distance from the beam to the location of each actor in the scene, for a total of 126 additional derived features. The intuition for the shifting windows computation is that, like the detection of the utterance, the beam location signal from the microphone array arrives at a certain (unknown) delay from the actual utterance, hence the most informative portion of the signal is in one of these windows.

Acoustic. This class includes a single stream feature that captures the amplitude of the audio signal at any point in time. Based on this stream feature we computed a derived feature which captures the average log energy throughout the actual utterance. In addition, we computed the average log energy (as well as a normalized version with respect to the entire utterance) in a set of windows that cover regions prior to and after the start, and prior to and after the end of the actual utterance. In total, 126 additional derived features were computed to capture the energy of the signal in different regions near the beginning and end of the utterance.

Vision. This class includes 18 stream features (6 per participant), which continuously capture vision information about the participants in the scene. The information includes location, size, and orientation of the tracked faces, the confidence score from the face tracker. In addition, two streams capture information about the inferred visual focus of attention of each participant at two different levels of granularity. One is based on a face detection algorithm and provides information about face orientation in 5 categories: frontal, left, slightly-left, right, slightly-right. The second captures a heuristic that the system uses to classify the face orientation into three categories: left, right, frontal. The information from the face location streams was used in conjunction with the information from the microphone array to construct a set of manually derived features. In addition, we constructed a large set of derived features based on the attentional streams, by computing the mean value the face was oriented in a certain position, throughout various windows prior to and after the beginning, and prior to and after the end of the actual utterance.

5. EXPERIMENTS AND RESULTS

The data collected in the user study described earlier was separated in 15 folds, corresponding to the 15 groups of 4 participants that interacted with the system. We randomly selected 8 of the 15 folds for training/development, and held the other 7 aside as a test set. We developed and analyzed the proposed methods by performing cross-validation experiments on the development set. Once the methods were developed and stable, we evaluated them by training the models on the development set and testing against the test set. For the binary *Target* and *Overlap* problems the training set contained 2101 utterances and the test set contained 2078 utterances. Because the *Source-2* and *Source-3* problems use only the interactions between the system and two or three other participants, respectively, the amount of data is reduced compared to the other two problems. Specifically, for the *Source-2* problem, the train set contained 1218 utterances and the test set contained 1133 utterances; for the *Source-3* problem, the train set contained 871 utterances and the test set contained 941 utterances.

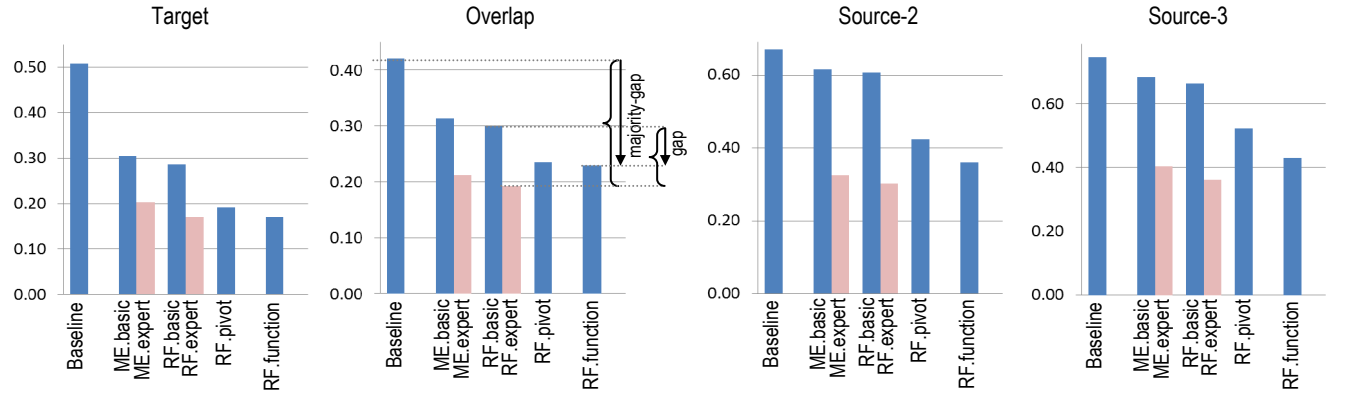

Figure 3. L2 test-set performance of the models on the four multimodal inference problems

The models for inferring speech source, target and overlap that we develop below are to be used in a spoken dialog system in conjunction with decision theoretic approaches for managing turn-taking actions [16]. As such, in addition to accuracy, it is important that these models produce well-calibrated probabilities. We therefore focus the evaluation on the L2-norm (mean squared error of the model's probabilistic output to 0-1 targets), which is a proper scoring rule, and reflects both model refinement and calibration [17]. For completeness, we also report classification error.

We begin by constructing models that only use the basic point features and models that use both the basic and the manually constructed derived features. We refer to the former as *basic* models and the latter as *expert* models. We experimented with maximum entropy models (ME) and random forests (RF). To regularize, the maximum entropy models used standard L1 regularization, with the regularization weight optimized in a 10-fold cross-validation procedure on the train set. For the random forest approach, we trained 200 trees, using information gain as the splitting criterion. We trained each tree with a bootstrap sample from the training set, and at each node randomly sampled 30% of all features to consider for potential splits[2].

The test set results for the *basic* and *expert* models, trained using the maximum entropy and the random forest approach, across all four problems, are shown in Table 1 and illustrated in Figure 3. In each case, the *expert* model improved performance on both L2 and classification error by a significant amount, highlighting the utility of the manually constructed derived features. The random forest approach consistently outperforms the maximum entropy models by a small amount. In contrast to maximum entropy models, random forests are capable of learning non-linear models. The results suggest non-linearity might be useful on these problems. The random forest approach also may be more robust to overfitting.

The *expert* random forest model, which uses manually crafted features, attains a classification error rate of 10.8% on the addressee identification problem (*Target*), and 12.3% on overlap detection. We believe this operating range can enable better decisions for multiparty turn-taking in spoken dialog systems. The error rates are somewhat higher in the Source-2 and Source-3 problems (19.5% and 24.6%, respectively). Recall however that these problems are more difficult because these are 4- and 5-way classification problems, we are trying to simultaneously identify both the source and overlaps, and we have less data to train on.

	Model	L2			Error
		Test	m-gap (%)	gap (%)	Test
Target	**Majority baseline**	**0.51**	**0.0**	**-**	**44.2**
	ME.basic	0.30	60.3	-16.3	21.2
	ME.expert	0.20	90.2	71.4	13.9
	RF.basic	**0.29**	**65.8**	**0.0**	**21.1**
	RF.expert	**0.17**	**100.0**	**100.0**	**10.8**
	RF.pivot	0.19	93.8	81.9	11.8
	RF.basis-function	0.17	100.0	99.9	10.0
Overlap	**Majority baseline**	**0.42**	**0.0**	**-**	**29.2**
	ME.basic	0.31	46.9	-13.6	21.8
	ME.expert	0.21	91.4	81.6	14.2
	RF.basic	**0.30**	**53.3**	**0.0**	**20.2**
	RF.expert	**0.19**	**100.0**	**100.0**	**12.3**
	RF.pivot	0.23	81.4	60.1	16.1
	RF.basis-function	0.23	83.4	64.4	15.4
Source-2	**Majority baseline**	**0.67**	**0.0**	**-**	**60.2**
	ME.basic	0.62	15.0	-3.0	52.9
	ME.expert	0.33	93.7	92.4	20.6
	RF.basic	**0.61**	**17.5**	**0.0**	**51.5**
	RF.expert	**0.30**	**100.0**	**100.0**	**19.5**
	RF.pivot	0.42	67.1	60.1	29.5
	RF.basis-function	0.36	84.7	81.4	23.9
Source-3	**Majority baseline**	**0.75**	**0.0**	**-**	**65.7**
	ME.basic	0.68	16.5	-6.3	60.9
	ME.expert	0.40	89.4	86.5	28.5
	RF.basic	**0.66**	**21.4**	**0.0**	**60.6**
	RF.expert	**0.36**	**100.0**	**100.0**	**24.6**
	RF.pivot	0.52	58.2	46.8	35.6
	RF.basis-function	0.43	81.8	76.9	28.8

Table 1. Test-set comparison of the models on the four multimodal inference problems. The majority baseline and the basic and expert models that define the gap scales are in bold; the best performing stream-based approach is highlighted

We now turn our attention to the proposed stream random forest models, which use the basic point features and directly leverage the temporally streaming features by using the *pivot* and *basis-function* approaches. Like before, we build 200 trees using bootstrap samples. For the *pivot* approach, we used the cosine similarity metric on windows in the 60 frame stream. To keep the number of windows manageable, we only considered windows that start or stop on every third frame, yielding a total of 210 possible windows. At each node, to create a split, we considered a

[2] To set parameters such as the number of trees in the random forest and the fraction of features to consider at each split, we conducted preliminary cross-validation experiments on the training set. Results indicated that 200 trees were sufficient to asymptote in performance and that the random forest was robust to a wide range of parameter values.

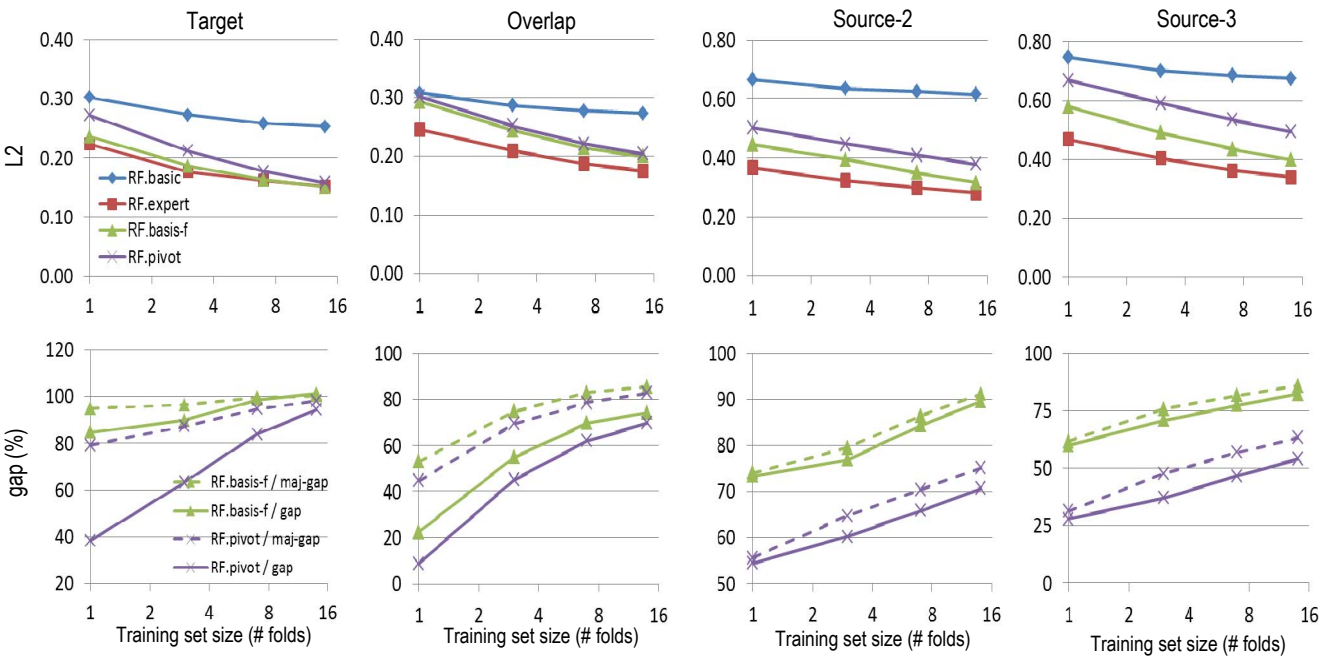

Figure 4. Effect of dataset size on random forest models across the four problems.
Top: L2 error versus number of training folds for basic, expert, pivot, and basis-function models.
Bottom: majority-gap and gap versus number of training folds for pivot and basis-function models.

randomly selected 30% of the point and stream features; for splits on stream features we considered 10% of the data-points that arrive at that node as possible pivots to search over, with 10% of all possible windows.

For the *basis-function* approach we initially considered the min, max, and the first, second, and third moment of the streams. Experiments conducted in cross-validation on the train set revealed the importance of using central moments for the *Overlap* problem: a very informative feature is the standard deviation of the microphone array beam. We therefore added the second and third *central* moments to the set of basis functions. Like in the pivot approach, the split construction algorithm had access to a randomly selected subset containing 30% of the point and stream features, and, in this case, 10% of the possible combinations of basis function and windows for stream feature splits.

We compare these models against the majority baseline, and against the basic and expert models previously constructed using the random forest approach (since they outperformed the maximum entropy models). To facilitate these comparisons, we compute two *gap* measures. The first, denoted *majority-gap* (m-gap in Table 1) indicates how much of the L2 gap between the *majority* baseline and the *expert* model is covered by a given approach (see also Figure 3): 0% indicates that an approach performs no better than the *majority* baseline and 100% indicates that the approach performs as well as the *expert* model, which uses the manually engineered features. The second measure, denoted simply *gap* indicates how much of the reduction in L2 error between the *basic* and *expert* models a given approach covers (see also Figure 3).

The results shown in Table 1 and illustrated in Figure 3 indicate that both the *pivot* and *basis-function* approaches significantly improve upon the *basic* model in all four problems; both methods leverage some of the information in the temporally streaming features. The results also indicate that the basis-function approach outperforms the pivot approach on each problem, improving L2 loss by as little as 0.005 on *Overlap* to as much as 0.09 on *Source-3*. This corresponds to gap scores 4% to 30% better for the basis-function approach.

Comparing the basis function approach to the *expert* model, we see gap scores that range from 64% to 100% and majority-gap scores that vary from 82% to 100%. These results indicate that the basis function random forest method performs almost as well as random forests with manually constructed features (but without requiring human expertise).

5.1 Learning rate experiments

The results discussed in the previous section were obtained by training models on the 8-fold development set and testing on the held-out test set – only half of the available data was used for training. We expect that, with more data, the stream-enabled random forest would further improve performance: as more data is available the trees can grow deeper and the model might capture increasingly more sophisticated cross-stream effects. To explore this hypothesis we performed another experiment where we used cross-validation on the entire dataset (all 15 folds) to train models with varying amounts of data. For each test fold, we randomly selected 1, 3, 7, and 14 of the remaining train folds and built models. To reduce variance in the estimates, we repeated this process 4 times for each test fold, and report the average of the performance of the 4 models.

The results are illustrated in Figure 4: the top row shows the performance of the *basic*, *expert*, *basis-function* and *pivot* random forest models across the four problems, as the amount of training data is increased from 1 to 3 to 7, and finally to 14 folds; the bottom row shows the corresponding gap scores. As is typical for reporting learning curves, the x-axis is shown on a log-scale. The plots indicate that, as more data becomes available, the gap and majority-gap scores increase: the methods learning from the raw streams are approaching the performance of the expert baseline. This suggests that the need to manually construct stream-based

features is mitigated as more data becomes available, allowing one to focus on collecting more data (always a good thing) instead of trying to understand the temporal stream data and manually construct effective derived features from them.

5.2 Alternate Methods and Future Work

There are other models that learn directly from sequence data. We have conducted preliminary experiments with both Hidden Markov Models (HMMs) and Recurrent Neural Nets (RNNs).

With HMMs, for each class we use Baum-Welch to train a separate model. Each HMM uses discrete observations derived via vector quantization from the parallel streams. The likelihood scores resulting from HMM decoding of the stream sequences are used as additional point features for learning (*i.e.*, one additional point feature is created for each class). There are many design decisions that must be made to train HMMs and prevent overfitting that must be investigated.

We are also experimenting with using RNNs to create point features that encode the information in the streams. So far, given the small training data, we have been unable to train RNNs successfully from all streams jointly. One potential solution is to train one RNN per stream to construct point features.

Overfitting is a major issue for these models given the amount of training data. Further investigation of these methods is part of future work.

6. CONCLUSION

We investigated discriminative learning approaches (maximum entropy and random forests) for building multimodal inference models for the problems of speaker, addressee, and overlap detection in human-computer interaction. Results indicate that both models perform well on these tasks, with the random forest approach consistently outperforming the maximum entropy method by a small amount.

In an effort to shortcut the manual feature construction process, we developed a novel extension to decision tree learning that enables trees to reason directly about temporally streaming features. The results suggest that the new approach can learn directly from the stream features. While the approach does not achieve the full performance of models trained on the features we manually engineered, it eliminates the often challenging process of manually constructing task-dependent features; results also suggest that adding more data narrows the gap between automatically and manually generated features.

We believe there is room to further improve the stream-based trees. For instance, while selecting stream windows in the pivot and basis-function approaches introduces information about the temporal dimension, the similarity metric and the aggregate statistics we used are not themselves time sensitive. An analysis of the constructed trees can shed more light on the type of within- and cross-stream feature representations that tree can create. An evaluation against features manually engineered by multiple experts on different problems would create a better understanding of the relative performance of this approach. We hope that with further development the stream-based tree construction methods would even be able to learn feature patterns and combinations that human experts might miss.

7. REFERENCES

[1] Bohus, D., and Horvitz, E., (2009). Dialog in the Open World: Platform and Applications, *ICMI'09*, Boston, MA

[2] Bohus D., and Horvitz, E., (2011). Multiparty Turn Taking in Situated Dialog: Study, Lessons, and Directions, *SIGdial'2011*, Portland, OR.

[3] Katzenmaier, R., Stiefelhagen, R., and Schultz, T. 2004. Identifying the addressee in human-human-robot interactions based on head pose and speech. In *Proceedings of ICMI'04*.

[4] Van Turnhout, K., Terken, J., Bakx, I., and Eggen, B. 2005. Identifying the intended addressee in mixed human-human and human-computer interaction from non-verbal features. In *Proceedings of ICMI'05*, 175-182.

[5] Jovanovic, N., Akker, R., and Nijholt, A. (2006). Addressee identification in face-to-face meetings. In *Proceedings of the EACL'06*, 169-176, 2006.

[6] Gatica-Perez, D., Lathoud, G., Odobez, J-M., McCowan, I. 2005. Multimodal multispeaker probabilistic tracking in meetings, In *Proceedings of ICMI'05*.

[7] Otsuka, K., Takemae, Y., and Yamato, J., 2005. A probabilistic inference of multiparty-conversation structure based on Markov-switching models of gaze patterns, head directions, and utterances. In *Proceedings of ICMI'05*.

[8] Imseng, D., and Friedland, G., 2009. Robust Speaker Diarization for Short Speech Recordings, In *Proceedings of ASRU'2009*

[9] Kadous, M.W. 1999. Learning comprehensible descriptions of multivariate time series. In *Proceedings of the International Conference on Machine Learning*.

[10] Morency, L.P. and de Kok, I. and Gratch, J. 2008. Context-based recognition during human interactions: Automatic feature selection and encoding dictionary. In *Proceedings of the 10th international conference on Multimodal interfaces*.

[11] Droppo, J. and Seltzer, M.L. and Acero, A. and Chiu, Y.H.B. 2008. Towards a non-parametric acoustic model: An acoustic decision tree for observation probability calculation. In *Ninth Annual Conference of the International Speech Communication Association*.

[12] Karimi, K. and Hamilton, H.J. 2010. Generation and Interpretation of Temporal Decision Rules. *Arxiv preprint arXiv:1004.3334*.

[13] Breiman, L. (2001). Random forests. *Machine Learning*, 45, 5-32.

[14] Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24, 123-140.

[15] Caruana, R., & Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. *ICML '06*, 161-168.

[16] Bohus, D., and Horvitz, E. (2011). Decisions about Turns in Multiparty Conversation: From Perception to Action. *ICMI'2011*, Alicante, Spain

[17] Cohen, I. and Goldszmidt, M., 2004. Properties and benefits of calibrated classifiers. In *Proceedings of EMCL/PKDD*. Pisa, Italy.