GRAPH-BASED LEARNING FOR PHONETIC CLASSIFICATION

Andrei Alexandrescu

University of Washington
Department of Computer Science
Seattle, WA, USA

Katrin Kirchhoff

University of Washington
Department of Electrical Engineering
Seattle, WA, USA

ABSTRACT

We introduce graph-based learning for acoustic-phonetic classification. In graph-based learning, training and test data points are jointly represented in a weighted undirected graph characterized by a weight matrix indicating similarities between different samples. Classification of test samples is achieved by label propagation over the entire graph. Although this learning technique is commonly applied in semi-supervised settings, we show how it can also be used as a post-processing step to a supervised classifier by imposing additional regularization constraints based on the underlying data manifold. We also present a technique to adapt graph-based learning to large datasets and evaluate our system on a vowel classification task. Our results show that graph-based learning improves significantly over state-of-the art baselines.

Index Terms— acoustic modeling, graph-based learning, classification, adaptation

1. INTRODUCTION

Acoustic modeling in automatic speech recognition (ASR) systems typically follows a well-established paradigm, that of hidden Markov models (HMMs) with Gaussian mixture (GM) based modeling of output distributions. In state-of-the-art systems, models typically undergo discriminative training and are adapted to test data using techniques such as MLLR [1] or MAP [2].

Several alternative or complementary approaches have been explored in the past, including different ways of modeling output distributions, such as Support Vector Machines (SVMs) [3] and neural networks [4], as well as novel training techniques, such as large-margin training [5]. However, the mainstream ASR community has been slow to adopt these (with some exceptions, such as [6, 7]), mainly because the standard methodology is well-tested, efficient, and easy to use, and also because new models or learning procedures may not scale well to large datasets.

Continued progress in ASR, however, does require exploring novel approaches, including new machine learning techniques, as well as adapting these to large data sets and the computational constraints that present-day ASR systems are subject to. In this paper we investigate graph-based learning as a way to improve over standard acoustic models.

Graph-based learning (not to be confused with graphical models) is a machine learning technique that jointly represents training and test data points as nodes in a graph. The graph edges are weighted with scores representing the similarity between data points. Based on this representation, several algorithms can be used to assign labels to the unlabeled (test) points, including spectral graph transducers (SGT) [8], random walks [9], and label propagation [10], each of which implements a particular classification function over the graph. The advantage shared by these approaches is that classification is based on a global consistency assumption: samples close to each other (in terms of the similarity function used in building the graph) should receive the same label. This applies to *both the labeled and the unlabeled samples*, i.e., the classification function exploits similarities between test samples in addition to similarities between test and training samples. In contrast, traditional systems (such as nearest-neighbor) only rely on similarity between the test and training samples. Graph-based learning is typically used in a semi-supervised, transductive setting where a relatively small amount of labeled data is used in conjunction with a large amount of unlabeled data. However, as we will show below, it can also be used as a postprocessing step applied to a standard supervised classifier trained on a large amount of labeled data and tested on a small amount of unseen data, which is the typical scenario in speech processing. In this case, graph-based learning provides a form of adaptation to the test data by constraining the decisions made by the first-pass classifier to accommodate the underlying structure of the test data.

One concern with graph-based learning in its original form is scalability. We present a sample merging technique that dramatically reduces the graph size and makes it essentially independent of the amount of training data, as well as a fast hyperparameter calibration technique that can be done offline (before the test data is seen) and works well. We present results on an 8-class vowel classifier with the goal of demonstrating the effect on speaker adaptation. Our approach improves significantly over state-of-the-art adaptation algorithms.

2. GRAPH-BASED LEARNING

Graph-based learning algorithms have been the focus of much recent machine learning research [11, 9, 12, 13]. In graph-based learning, data points ($1, \ldots, l$ labeled points and $l+1, \ldots, n$ unlabeled points) are arranged in a weighted undirected graph. The graph is characterized by a weight matrix W, whose elements $W_{ij} \geq 0$ are similarity measures between vertices i and j, and by its initial label vector $Y_L = (y_1, \ldots y_l), y_i \in \{1, \ldots, C\}$, that defines labels for the first l points. If there is no edge linking nodes i and j, $W_{ij} = 0$. Other than that, applications have considerable freedom in choosing the edge set and the W_{ij} weights. For example, a matrix can be defined as $W_{ij} = 1$ if x_i and x_j fall within each other's k nearest-neighbors (and zero otherwise). Another commonly-used weight matrix is defined by a Gaussian kernel:

$$W_{ij} = \exp\left[-\frac{d(x_i, x_j)^2}{\alpha^2}\right] \quad (1)$$

where $d(x_i, x_j)$ is the (estimated) distance between feature vectors x_i and x_j and α is a bandwidth hyperparameter. Various distance measures can been used, e.g. Cosine distance, Euclidean distance, Jeffries-Matusita distance, or Jensen-Shannon divergence. The distance measure determines the graph construction and is the most important factor in successfully applying graph-based learning.

Learning algorithms defined on graphs include e.g. Blum and Chawla's mincut algorithm [11], Szummer and Jaakkola's random walk-based approach [9], and label propagation [10]. In this work, we use the latter.

2.1. Label Propagation

Label propagation [10] is a representative graph-based learning algorithm and operates in a semi-supervised manner by taking advantage of both training and test samples during learning. Once W is constructed, the basic label propagation algorithm proceeds as follows:

1. Initialize the matrix P as $P_{ij} = \frac{W_{ij}}{\sum_j W_{ij}}$
2. Initialize a $n \times C$ matrix f with binary vectors encoding the known labels for the first l rows: $f_i = \delta_C(y_i)\ \forall i \in \{1, 2, \ldots, l\}$ (the remaining rows of f can be zero)
3. $f' \leftarrow P \times f$
4. Clamp already-labeled data rows: $f'_i = \delta_C(y_i)\ \forall i \in \{1, 2, \ldots, l\}$
5. If $f' \cong f$, stop
6. $f \leftarrow f'$
7. Repeat from step 3

The f matrix contains the solution in rows $l+1$ to n in the form of label probability distributions. Figure 1 shows a graph before and after the label propagation process.

Many applications need hard labels, obtainable by:

$$\hat{y}_i = \arg\max_j f_{ij}\ \ \forall i \in \{l+1, \ldots, n\} \quad (2)$$

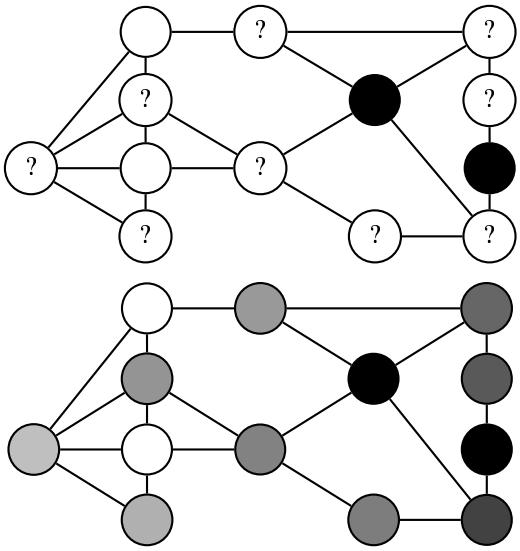

Fig. 1. A graph before and after label propagation. The labels are encoded as white and black, and all edges have unit weight. The process fills the unlabeled nodes with shades of gray, depending on the connectivity with neighboring nodes. Nodes are filled even when they don't directly connect with any of the labeled nodes.

It is easy to prove [14] that the algorithm converges to the analytic solution:

$$f_U = (I - P_{UU})^{-1} P_{UL} Y_L \quad (3)$$

where P_{UL} is P's submatrix connecting unlabeled to labeled samples (rows $l+1$ to n and columns 1 to l), P_{UU} is P's submatrix connecting unlabeled samples with one another (rows $l+1$ to n and columns $l+1$ to n), and Y_L is the $l \times C$ matrix encoding the known labels: $Y_{L(\text{row } i)} = \delta_C(y_i)$. Practical applications might still implement the iterative variant as it does not require matrix inversion.

The fixed point of the label propagation algorithm minimizes the following cost function:

$$S = \sum_{i>n \vee j>n} W_{ij}(\hat{y}_j - \hat{y}_i)^2 \quad (4)$$

This cost function has an obvious minimum when all samples are assigned the same label, so an additional restriction is that the estimated labels for the known data points must coincide with their actual labels:

$$\hat{y}_i = y_i\ \ \forall i \in \{1, \ldots, l\} \quad (5)$$

S measures the extent to which the labeling allows large weight edges to link nodes of different labels. The label propagation algorithm finds a labeling that, to the extent possible, assigns identical labels to nodes linked by high weights.

Label propagation has been used successfully for various classification tasks, e.g. image classification and handwritten

digit recognition [14]. In natural language processing, LP has been used for document classification [14], word sense disambiguation [15], and sentiment categorization [16].

3. GRAPH-BASED ACOUSTIC CLASSIFICATION

Applying graph-based learning to acoustic classification raises unique challenges:

- *Similarity measure:* As mentioned, choosing an appropriate similarity measure is key to graph-based learning. It is unclear what similarity measure would be optimal in acoustic feature spaces.
- *Scalability:* Acoustic data is typically available in large quantities. In its basic setup as described above, a graph is constructed in memory for the entire training data and the test data at once, which is infeasible for all but very small databases.
- *Adaptation:* Originally, graph-based learning was formulated for semi-supervised scenarios, where a large amount of unlabeled but a small amount of labeled data are present. In typical speech processing applications, we find the opposite situation. In these cases, graph-based learning can still be of benefit due to the global consistency assumption it enforces, thus effectively implementing a form of test data adaptation. However, this requires changes to the basic algorithm.

The following subsections describe in detail how our system addresses these challenges.

3.1. Similarity Measure

Most work in graph-based learning has used standard distance measures such as Cosine or Euclidean distance.

A data-driven method presented in previous work [17] defines distances on the outputs of a first-pass classifier. The resulting feature space consists of probability distributions over the desired classes, and probability divergence measures with well-understood statistical properties can thus be used as distance measures. Our experiments use a neural network with softmax output, trained on the original MFCC features, as the first-pass classifier. We use the Jensen-Shannon divergence as a distance measure:

$$d_{\mathrm{JS}}(Q_1, Q_2) = \frac{1}{2}\left[D_{\mathrm{KL}}(Q_1 \| M) + D_{\mathrm{KL}}(Q_2 \| M)\right] \quad (6)$$

where Q_1, Q_2 are probability distributions, $M = \frac{Q_1+Q_2}{2}$, and $D_{\mathrm{KL}}(Q_1 \| Q_2)$ is the Kullback-Leibler divergence:

$$D_{\mathrm{KL}}(Q_1 \| Q_2) = \sum_i Q_1(i) \log \frac{Q_1(i)}{Q_2(i)} \quad (7)$$

If in (7) we define $\log \frac{0}{0} = 0$, then the Jensen-Shannon divergence is well-defined, continuous, and bounded.

The distance d_{JS} is converted to a similarity measure by using a Gaussian kernel of parameterized width (1).

3.1.1. Hyperparameter Optimization

The quality of the similarity measure hinges not only on choosing a good distance measure, but also on finding a good value for the hyperparameter α in (1). Choosing the optimal α is an open research question; several heuristic methods have been used in practice. In [14], α is optimized to yield a labeling of minimum entropy, subject to the constraint that the labeling must respect the labels of the training set. However, optimizing α by using gradient descent for each utterance adds considerable overhead to the classification time.

We propose an efficient method of calibrating α that works offline (only uses the training data) and is inspired by maximum margin techniques. First, we compute the average intra-class distance (d_{intra}) and inter-class distance (d_{inter}):

$$d_{\mathrm{intra}} = \frac{\sum_{i,j:i\neq j, y_i = y_j} d(x_i, x_j)}{N_{\mathrm{intra}}}; \quad d_{\mathrm{inter}} = \frac{\sum_{i,j:\ i\neq j, y_i \neq y_j} d(x_i, x_j)}{N_{\mathrm{inter}}} \quad (8)$$

where N_{intra} and N_{inter} are the counts of the respective terms. Ideally, $d_{\mathrm{intra}} > d_{\mathrm{inter}}$ by a large margin, otherwise the data has poor separability. We then choose α such that two samples distanced at $\frac{d_{\mathrm{inter}}+d_{\mathrm{intra}}}{2}$ have a similarity of 0.5:

$$\exp\left[-\frac{(d_{\mathrm{intra}} + d_{\mathrm{inter}})^2}{4\alpha^2}\right] = \frac{1}{2} \Rightarrow \alpha = \frac{d_{\mathrm{intra}} + d_{\mathrm{inter}}}{2\sqrt{\ln 2}} \quad (9)$$

The intuition behind this choice is that, given that similarity has range $[0, 1]$, two samples placed at the most ambiguous distance should be midway in terms of similarity as well.

Computing the average distances d_{intra} and d_{inter} would necessitate $O(l^2)$ distance computations, one for each pair of training samples. A time-efficient approach we choose in practice is to do a random sampling: two samples x_i and x_j are randomly chosen from the training set, their distance is computed and considered for d_{intra}, if $y_i = y_j$, or for d_{inter} otherwise. We used 2.5% of the data in five successive trials; α varied by no more than 1% among trials. Choosing α in this manner yielded much better performance in our tests than grid search and a method based on the minimum spanning tree [14, § 7.3].

3.1.2. Interpolation with Prior Distributions

For the training set we have access to the true labels and consequently to the sample prior probability distributions:

$$P_p^{y_i} = (0_1, \ldots, 0_{y_i-1}, 1_{y_i}, 0_{y_i+1}, \ldots, 0_C) = \delta_C(y_i) \quad (10)$$

($\delta_C(l)$ denotes a Kronecker vector of length C with 1 in the l^{th} position and 0 elsewhere.) These prior distributions represent the ground truth, so they are highly informative for classification. Using them exclusively, however, would lose smoothness information, so they should best be used in interpolation with the soft predictions resulting from the first-pass classifier running against its own training data. We chose an equal-weight interpolation $\frac{P+P_p}{2}$ throughout our experiments.

3.2. Scalability

The drawback of graph-based learning is its computational inefficiency: typically, the graph is constructed on-demand for the joint training and test set, which requires large memory and many CPU operations. However, a dramatic reduction in the number of nodes is possible without a loss in precision. A simple algorithm (outlined below) reduces the number of labeled nodes from l to C without impacting the final result. The size of the graph becomes essentially independent from the amount of training data used. The intuition behind the reduction process is that labeled nodes having the same label can be "collapsed" together because their identity does not matter.

Proposition. Consider the matrices P (size $n \times n$) and f (size $n \times C$) initialized for the label propagation iterative algorithm (§ 2.1). Define the matrix $R(a,b)$ (where $1 \leq a < b \leq l$) of size $(n-1) \times (n-1)$ obtained from P by adding the a^{th} column to the b^{th} column, followed by eliminating the a^{th} row and a^{th} column:

$$R(a,b) = \begin{bmatrix} P_{1,1} & \cdots & P_{1,a-1} & P_{1,a+1} & \cdots & P_{1,a}+P_{1,b} & \cdots \\ \cdots & & & & & & \\ P_{a-1,1} & \cdots & P_{a-1,a-1} & P_{a-1,a+1} & \cdots & P_{a-1,a}+P_{a-1,b} & \cdots \\ P_{a+1,1} & \cdots & P_{a+1,a-1} & P_{a+1,a+1} & \cdots & P_{a+1,a}+P_{a+1,b} & \cdots \\ \cdots & & & & & & \\ P_{n,1} & \cdots & P_{n,a-1} & P_{n,a+1} & \cdots & P_{n,a}+P_{n,b} & \cdots \end{bmatrix} \tag{11}$$

Define the matrix $g(a)$ obtained by eliminating f's a^{th} row:

$$g(a) = \begin{bmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,C} \\ \cdots & & & \\ f_{a-1,1} & f_{a-1,2} & \cdots & f_{a-1,C} \\ f_{a+1,1} & f_{a+1,2} & \cdots & f_{a+1,C} \\ \cdots & & & \\ f_{n,1} & \cdots & & f_{n,C} \end{bmatrix} \tag{12}$$

If the rows a and b of f are identical, then using $R(a,b)$ and $g(a)$ for label propagation yields the same label predictions (the bottom $n-l$ rows of g) as using P and f.

Proof. Consider the iterative step $f' \leftarrow P \times f$. The element $f'_{k,j}$ is:

$$f'_{k,j} = \sum_{i=1}^{n} P_{k,i} f_{i,j} = P_{k,a} f_{a,j} + P_{k,b} f_{b,j} + \sum_{\substack{i=1 \\ i \notin \{a,b\}}}^{n} P_{k,i} f_{i,j} \tag{13}$$

But $f_{a,j} = f_{b,j}$ by the hypothesis, therefore:

$$f'_{k,j} = (P_{k,a} + P_{k,b}) f_{b,j} + \sum_{\substack{i=1 \\ i \notin \{a,b\}}}^{n} P_{k,i} f_{i,j} \tag{14}$$

It can be easily verified by inspection that $f'_{k,j} = g'(a)_{k,j}$ for $1 < k < a$ and $f'_{k,j} = g'(a)_{k-1,j}$ for $a < k \leq n$, where $g'(a) = R(a,b) \times g(a)$. As $a < l$ (by the hypothesis) and the result of the algorithm is in rows $l+1$ to n, it follows by induction that the iteration with $R(a,b)$ and $g'(a)$ will yield identical labelings as iterating with P and f'. □

This means that the label propagation problem can be reduced by one labeled sample whenever there are two labeled samples having he same label. Applying the reduction process iteratively, we obtain a reduced matrix R^{min} of size $(C+n-l) \times (C+n-l)$ and a reduced label matrix g^{min} of size $(C+n-l) \times C$, an important reduction in size. The process of reduction requires only $O(n(n-l))$ additions and does not require additional memory. In practice, starting with the large matrix P is not even necessary; the matrix R^{min} can be constructed in-place by accumulation.

After reduction, assuming (without loss of generality) that labeled samples are ordered in increasing order of their labels, the iteration formula becomes considerably simpler:

$$f_U \leftarrow P_{UU} f_U + R^{\text{min}}_{UL} \tag{15}$$

The matrix f_U can be stored in column-major order to render the multiplication cache-friendly. The P_{UU} matrix grows with the square of unlabeled data count, growth controllable by using a sparse structure that ignores distances beyond a limit or cuts off the number of neighbors. Our data set had test utterances short enough to allow affording a dense P_{UU} matrix.

3.3. Adaptation

Adaptation is an important challenge in speaker-independent ASR systems. Label propagation is inherently adaptive because it uses the self-similarity of the test data in addition to the similarity of the test data with the training data. To properly exploit the adaptive nature of label propagation, we operate a simple but very helpful change to the matrix W.

Most graph-based learners represent each sample (train or test) as a vertex. In classifying a test utterance using a graph, that utterance's group of vertices is much less numerous than the training vertices. As such, the accumulated similarity with the training samples will dwarf the self-similarity of the cluster; effectively, classification tends to approximate an unsophisticated nearest-neighbor technique. This is because in the random walk interpretation, the probability that a random walk jumps straight to the closest labeled vertex is much greater than the probability of the random walk exploring neighboring unlabeled samples.

To benefit of adaptation, we want to manipulate the density of the graph in the region of the test utterance. We achieve this by adjusting W_{ij} linking unlabeled samples with one another by:

$$W_{ij} \leftarrow \frac{l}{n-l} W_{ij} \quad \forall\, i > l, j > l \tag{16}$$

This artificially simulates that there are as many test as train samples, greatly enhancing the adaptive properties of the algorithm. Although simple, this adjustment is extremely effective; without it, the classification degenerates in nearest-neighbor (i.e., the algorithm in § 2.1 converges in exactly one step) and never improves upon the first-pass classifier.

4. DATA

We performed initial exploratory experiments on an 8-vowel classification task collected for the Vocal Joystick (VJ) project [18], whose goal it is to develop voice-controlled assistive devices for individuals with motor impairments. In the typical setup, a VJ user can exercise analog, continuous control over mouse cursor movements by using vowel quality, pitch, or loudness. One of the components of the VJ system is a speaker-independent vowel classifier whose output is used to control, for example, the direction in which a mouse cursor moves. In this and similar scenarios, phonetic classification that is robust against speaker variation is of utmost importance in order to avoid rejection of the system by the user due to inaccurate recognition of control commands.

For training this classifier, a corpus was collected consisting of 11 hours of recorded data of which we selected a subset. The sizes of the train, development, and test data are shown in Table 1.

Table 1. Training, development, and testing data used in the Vocal Joystick experiments.

Set	Speakers	Samples	Non-silent audio
Training	21	$420 \cdot 10^3$	1.16h
Development	4	$200 \cdot 10^3$	0.56h
Test	10	$80 \cdot 10^3$	0.22h

This scenario is a good test bed for our proposed approach since an already tuned, high-performing baseline system with standard adaptation methods exists for this data set. In addition, the focus on phonetic classification allows us to focus on the acoustic models while ignoring e.g. language model and search effects that would characterize large-vocabulary systems. At the same time, this corpus is vastly more realistic than the toy tasks used in machine learning since it contains hundreds of thousands of samples.

5. EXPERIMENTS AND RESULTS

We tested our two-pass system by directly using the outputs of the best classifier on the VJ corpus to date, created by Li [19]. Li's classifier is a multi-layer perceptron (MLP) enhanced with a regularized adaptation algorithm. The adaptation algorithm uses a regularizer that prevents the regularized model diverging too much from the unadapted system, thus avoiding overtraining on adaptation data. We used the same MLP (50 hidden units and a window size of 7 samples) and the same adaptation algorithm as Li.

We apply our system to both the non-adapted MLP outputs and the adapted outputs. In each case, a graph (of reduced size using the result of Proposition 1) was built for each test utterance, after which iterative label propagation was applied to the graph. As an additional baseline we use GMMs (a) without adaptation and (b) with MLLR adaptation. The adaptation experiments used 5-fold cross-validation, each time using a held-out part of the test data for computing adaptation parameters. The results are shown in Table 2. Boldface numbers are significantly better than the comparable baselines.

Table 2. Error rates (means and standard deviations over all speakers) using a Gaussian Mixture Model (GMM), multilayer perceptron (MLP), and MLP followed by a graph-based learner (GBL), with and without adaptation. The highlighted entries represent the best error rate by a significant margin ($p < 0.001$).

Model	Error Rate (%)	
	Dev	Test
GMM, no adaptation	n/a	39.62
MLP, no adaptation	24.81±10.69	31.91±9.39
MLP+GBL, no adaptation	**21.91±10.52**	**28.75±12.31**
GMM+adaptation	n/a	20.05±3.76
MLP+adaptation	n/a	12.18±3.51
MLP+adaptation+GBL	n/a	**8.32±3.21**

The similarity of choice was Jensen-Shannon divergence, which is theoretically motivated due to operation in probability space; to confirm that it is a good-quality distance, we compared it with dev set performance for two commonly-used distance measures: Cosine distance and Euclidean distance. They both engendered higher error rates (22.62±11.23% for Cosine and 22.48±11.00% for Euclidean).

6. DISCUSSION

We have shown that a graph-based learner applied to the outputs of a first-pass classifier significantly improves classification results on an 8-vowel discrimination task. This holds for both unadapted and adapted classifiers, i.e., the improvement is additive to standard adaptation methods. In addition, we have shown how graph-based learning can be used on large data sets. The adaptation achieved by the graph-based learner can be described within the framework of manifold regularization [20], which formulates the optimal classifier as the function:

$$F = \arg\min_{\mathbf{f}} \left[\sum_{i=1}^{l} c(y_i, f(x_i)) + \lambda_1 \|f\|^2 + \lambda_2 f_{LU}^T L f_{LU} \right] \quad (17)$$

where c is a loss function describing the basic performance of the classification function f, $\|f\|$ denotes parameter regularization of the function, and the third term enforces smoothness of f over the graph (the global consistency assumption). The λ's are tunable weights. In our system described above,

the graph-based learner regularizes the first-pass classifier by enforcing constraints corresponding to the third term in (17): outputs are required to respect the underlying data manifold. The first term corresponds to the basic training criterion for the MLP and the second term to Li's adaptive parameter regularization. The difference is that the graph-based learner was tuned independently of the MLP and its parameter regularization instead of jointly. We intend to extend our current framework to joint optimization in the future. Previous work on adaptation in ASR has either ignored regularization or has limited itself to parameter regularization (eg the regularized MLLR algorithm). Manifold regularization is an important new extension to current methods.

Further future work will include application to more complex tasks, such as classification of more complex phone sets, and application to ASR tasks.

Acknowledgments

This work was supported by NSF grants IIS-032676 and IIS-0326382, and DARPA Contract No. HR0011-06-C-0023. Any opinions, findings, and conclusions or recommendations expressed in this material belong to the authors and do not necessarily reflect the views of these agencies. We thank Jon Malkin, Amar Subramanya, and Xiao Li for help with the baseline system.

7. REFERENCES

[1] M. Gales and P.C. Woodland, "Mean and variance adaptation within the MLLR framework," *Computer, Speech and Language*, 1996.

[2] J.L. Gauvain and C.H. Lee, "Maximum a-posteriori estimation for multivariate gaussian mixture observations of markov chains," *IEEE Transactions on Speech and Audio Processing*, vol. 2, pp. 291–298, 1994.

[3] A. Ganapathiraju and J. Picone, "Hybrid SVM/HMM architectures for speech recognition," in *Proceedings of NIPS*, 2000.

[4] H. Bourlard and N. Morgan, *Connectionist Speech Recognition: A Hybrid Approach*, Kluwer Academic Publishers, 1994.

[5] F. Sha and L. Saul, "Large margin gaussian mixture modeling for phonetic classification and recognition," in *Proceedings of ICASSP*, 2006, pp. 265–268.

[6] Q. Zhu, A. Stolcke, B. Chen, and N. Morgan, "Using MLP features in SRI's conversational speech recognition system," in *Proceedings of Eurospeech*, 2005, pp. 2141–2144.

[7] A. Stolcke, F. Grezl, M-Y Hwang, X. Lei, N. Morgan, and D. Vergyri, "Cross-domain and cross-language portability of acoustic features estimated by multilayer perceptrons," in *Proceedings of ICASSP*, 2006, pp. 321–324.

[8] T. Joachims, "Transductive learning via spectral graph partitioning," in *Proceedings of ICML*, 2003.

[9] Martin Szummer and Tommi Jaakkola, "Partially labeled classification with markov random walks," in *Advances in Neural Information Processing Systems*, 2001, vol. 14, `http://ai.mit.edu/people/szummer/`.

[10] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," Tech. Rep., CMU-CALD-02, 2002.

[11] A. Blum and S. Chawla, "Learning from labeled and unlabeled data using graph mincuts," *Proc. 18th International Conf. on Machine Learning*, pp. 19–26, 2001.

[12] X. Zhu and Z. Ghahramani, "Towards semi-supervised classification with Markov random fields," Tech. Rep., Technical Report CMU-CALD-02-106). Carnegie Mellon University, 2002.

[13] Avrim Blum, John Lafferty, Mugizi Robert Rwebangira, and Rajashekar Reddy, "Semi-supervised learning using randomized mincuts," in *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, New York, NY, USA, 2004, p. 13, ACM Press.

[14] Xiaojin Zhu, *Semi-Supervised Learning with Graphs*, Ph.D. thesis, Carnegie Mellon University, 2005, CMU-LTI-05-192.

[15] Zheng-Yu Niu et al, "Word sense disambiguation using label propagation based semi-supervised learning method," in *Proceedings of ACL*, 2005, pp. 395–402.

[16] A. Goldberg and J. Zhu, "Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization," in *HLT-NAACL Workshop on Graph-based Algorithms for Natural Language Processing*, 2006.

[17] Andrei Alexandrescu and Katrin Kirchhoff, "Data-Driven Graph Construction for Semi-Supervised Graph-Based Learning in NLP," in *HLT*, 2007.

[18] K. Kilanski, J. Malkin, X. Li, R. Wright, and J. Bilmes, "The Vocal Joystick data collection effort and vowel corpus," in *Interspeech*, September 2006.

[19] Xiao Li, *Regularized Adaptation: Theory, Algorithms and Applications*, Ph.D. thesis, University of Washington, 2007.

[20] M. Belkin, P. Niyogi, and V. Sindhwani, "On Manifold Regularization," *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTAT 2005)*, 2005.