Classification and Ranking Approaches to Discriminative Language Modeling for ASR

Erinç Dikici, *Student Member, IEEE*, Murat Semerci, *Student Member, IEEE*, Murat Saraçlar, *Member, IEEE*, and Ethem Alpaydın, *Senior Member, IEEE*

***Abstract*—Discriminative language modeling (DLM) is a feature-based approach that is used as an error-correcting step after hypothesis generation in automatic speech recognition (ASR). We formulate this both as a classification and a ranking problem and employ the perceptron, the margin infused relaxed algorithm (MIRA) and the support vector machine (SVM). To decrease training complexity, we try count-based thresholding for feature selection and data sampling from the list of hypotheses. On a Turkish morphology based feature set we examine the use of first and higher order n-grams and present an extensive analysis on the complexity and accuracy of the models with an emphasis on statistical significance. We find that we can save significantly from computation by feature selection and data sampling, without significant loss in accuracy. Using the MIRA or SVM does not lead to any further improvement over the perceptron but the use of ranking as opposed to classification leads to a 0.4% reduction in word error rate (WER) which is statistically significant.**

***Index Terms*—Discriminative language modeling (DLM), feature selection, data sampling, language modeling, ranking perceptron, ranking support vector machine (SVM), margin infused relaxed algorithm (MIRA), ranking MIRA, speech recognition.**

I. Introduction

GIVEN an acoustic speech signal, an automatic speech recognition (ASR) system generates multiple hypotheses (possible transcriptions), either in the form of a lattice or an N-best list. The hypotheses in the list are ordered with respect to their recognition scores, but the hypothesis having the highest score is not necessarily the most accurate transcription. Discriminative language modeling (DLM) was proposed as a post-processing step to determine the most accurate hypothesis from this list by considering other linguistic factors besides the recognition score.

Manuscript received February 24, 2012; revised July 16, 2012; accepted September 05, 2012. Date of publication October 02, 2012; date of current version December 10, 2012. This work was supported in part by TÜBİTAK under Project numbers 109E142, 109E186, and by the Turkish State Planning Organization (DPT) under the TAM Project number 2007K120610. The work of M. Saraçlar was supported by the TÜBA GEBTÜBİTAKP Award. The numerical calculations reported in this paper were performed at TÜBİTAK ULAKBİM, High Performance and Grid Computing Center (TR-Grid e-Infrastructure). Parts of this study were presented in the conference [1]. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Gokhan Tur.

E. Dikici and M. Saraçlar are with the Department of Electrical and Electronics Engineering, Boğaziçi University, 34342 Istanbul, Turkey (e-mail: erinc.dikici@boun.edu.tr; murat.saraclar@boun.edu.tr).

M. Semerci and E. Alpaydın are with the Department of Computer Engineering, Boğaziçi University, 34342 Istanbul, Turkey (e-mail: murat.semerci@boun.edu.tr; alpaydin@boun.edu.tr).

Digital Object Identifier 10.1109/TASL.2012.2221461

Classical DLM approaches define this as a classification problem where the aim is to discriminate the best possible transcription from the others while optimizing an objective function that is directly related to the word error rate (WER). But the hypotheses that are not the best transcription are not all equally bad, and the first of our contributions in this paper is to formulate this as a ranking problem which promises to be more informative [2]. In such a formulation, each hypothesis has a rank based on an accuracy measure, e.g., the number of word errors, and the aim is to try to reorder these in such a way that more accurate hypotheses are pushed towards the top of the list, as opposed to simply separating the best from all the bad ones.

A common model used for discriminative language modeling is the linear perceptron [3]. The margin infused relaxed algorithm (MIRA) updates the parameter of a linear model by enforcing a margin over multiple prototypes where one prototype is trained for each class [4]. The support vector machine (SVM) with the linear kernel is also a linear model but the formulation uses a convex program that can be solved optimally [5]. The second line of this paper is to compare these three models both for classification and ranking.

Data sets used in speech recognition are very large and for each instance, discriminative training uses the hypotheses in the N-best list. As the third direction, we investigate whether we can decrease complexity without loss in accuracy. For this, we try two possibilities: (1) With the high-order n-grams that we use as the input features, most will appear rarely in training and would not be informative; a simple count-based thresholding prunes the rare combinations, decreases the input dimensionality considerably and hence the training complexity of the learner that follows it. (2) In the N-best list, it is possible that most of the hypotheses will be very similar and the idea is that we can get the same discriminative power by using a small subset from the list, which will lead to significant saving from computation especially in the case of ranking.

In our previous work [1], we compared perceptron and SVM algorithms for ranking and proposed several data sampling and feature selection approaches for reranking ASR outputs in a Turkish broadcast news transcription task. In this paper, we extend these earlier results by including the SVM classifier and classification and ranking versions of the MIRA algorithm for completeness. We also increase the feature space by incorporating higher order n-grams, present the relationship between ranking versions of perceptron and SVM, and give a thorough statistical analysis and comparison of the results.

This paper is organized as follows: In Section II, we discuss the methods used in this study and give a brief literature review.

The experimental setup and results are given in Section III, which is followed by their discussion in Section IV. Section V concludes and contains some possible future work.

II. Methods

A. Discriminative Language Modeling Using Linear Models

The DLM approach can be viewed as a complementary method to baseline generative language modeling. In a DLM setup, the aim is to distinguish the good examples of an ASR output from the bad ones, while trying to optimize an objective function that is directly related to the word error rate (WER). The parameters of such a model are estimated in a discriminative setting.

DLM setup uses the outputs of an ASR system as its training examples. This set, ordered with respect to the generative recognition score, is called the N-best list. Each example of the training set is represented by a feature vector of the acoustic input, x, and the candidate hypothesis, y, denoted by $\mathbf{\Phi}(x, y)$. Each feature is associated with a weight representing the contribution of that feature and the model itself is defined by the associated weight vector, $\mathbf{w}$. In testing, this vector is used to select the highest scoring hypothesis: $y^* = \text{argmax}_y \langle \mathbf{w}, \mathbf{\Phi}(x, y) \rangle$. Learning here corresponds to optimizing the weight vector $\mathbf{w}$ and this can be defined as a classification or ranking problem.

Discriminative estimation of language models has been around for over ten years. The linear model framework we use in this study was first outlined in [6]. Feature-based structure of the linear model eases incorporating syntactic, semantic, morphological and n-gram information sources in a single mathematical representation [7], [8].

B. Classification Algorithms

1) Perceptron: The perceptron algorithm used in DLM is a multi-class perceptron variant used for structured prediction [3]. Given x_i, the acoustic input of utterance i and H_i, the list of all possible hypotheses (N-best list) for this utterance, the aim is to pick a hypothesis out of the set defined by H_i. Let us define the gold standard (y_i) as the hypothesis having the lowest WER, i.e., the oracle—if there are more than one such, the one having the highest recognition score is used. The current best is chosen as

$$z_i = \underset{z \in H_i}{\text{argmax}} \langle \mathbf{w}, \mathbf{\Phi}(x_i, z) \rangle \tag{1}$$

The idea of the perceptron algorithm is to reward features associated with the gold-standard and to penalize features associated with the current best:

$$\mathbf{w} = \mathbf{w} + \mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, z_i) \tag{2}$$

The algorithm requires several epochs (passes) over the training set but it has been shown that three epochs are adequate for practical purposes [3]. In the *averaged perceptron* which we also use in our experiments, once training is completed, the weights are averaged over the number of utterances (I) and epochs (T) to increase model robustness:

$$\mathbf{w}_{\text{avg}} = \frac{1}{IT} \sum_{i,t} \mathbf{w}_i^t \tag{3}$$

The perceptron algorithm in a classification setting is one of the most studied methods to estimate the parameters of the linear model [3], [9], [10].

2) Margin Infused Relaxed Algorithm (MIRA): The MIRA [4] trains a prototype for each class such that the dot product of an instance with the prototype belonging to its class, $\langle \mathbf{w}_{c_i}, \mathbf{\Phi}(x_i, y_i) \rangle$, is higher than the dot product with any other class prototype, $\langle \mathbf{w}_{\bar{c}_i}, \mathbf{\Phi}(x_i, y_i) \rangle$, where c_i is the class of $\mathbf{\Phi}(x_i, y_i)$, $\mathbf{w}_{c_i}$ is its class prototype (weight vector), and $\mathbf{w}_{\bar{c}_i}$ are the prototypes of other classes. The margin is defined as the minimum difference between the dot products and the aim is to train a classifier with a large margin.

For a two-class problem with $c_i \in \{\pm 1\}$, the binary MIRA iteratively updates a single prototype $\mathbf{w}$, just like the perceptron.

$$\mathbf{w} = \mathbf{w} + \tau_i c_i \mathbf{\Phi}(x_i, y_i) \tag{4}$$

Here the learning rates τ_i are hypothesis-specific, and are found by solving the following optimization problem:

$$\begin{aligned} \min_{\tau_i} \quad & \| \mathbf{\Phi}(x_i, y_i) \|^2 \tau_i^2 + 2 c_i \tau_i \langle \mathbf{w}, \mathbf{\Phi}(x_i, y_i) \rangle \\ \text{s.t.} \quad & 0 \leq \tau_i \leq 1 \end{aligned} \tag{5}$$

which gives

$$\tau_i = G\left(-\frac{c_i \langle \mathbf{w}, \mathbf{\Phi}(x_i, y_i) \rangle}{\| \mathbf{\Phi}(x_i, y_i) \|^2} \right) \tag{6}$$

The function $G(\cdot)$ determines how much to update the prototype if it misclassifies the instance.

$$G(u) = \begin{cases} 0 & u < 0 \\ u & 0 \leq u \leq 1 \\ 1 & 1 < u \end{cases} \tag{7}$$

Note that binary MIRA cannot be directly applied to our problem since we do not have the true labels of the instances; we predict the best hypothesis among many (N-best list) candidates. We propose single and multiple update versions of MIRA for structured prediction, similar to those proposed in [11] and [12].

The single update version updates only when the current best z_i is not the oracle y_i:

$$\mathbf{w} = \mathbf{w} + \tau_i^s (\mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, z_i)) \tag{8}$$

$$\tau_i^s = G^s \left(-\frac{\langle \mathbf{w}, \mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, z_i) \rangle}{\| \mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, z_i) \|^2} \right) \tag{9}$$

$$G^s(u) = \begin{cases} 0 & u < 0 \\ u & \text{otherwise} \end{cases} \tag{10}$$

The multiple update version scans over pairs of the oracle y_i and all other hypotheses $y_k \in H_i$, and updates as:

$$\mathbf{w} = \mathbf{w} + \tau_k^m (\mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_k, y_k)) \tag{11}$$

$$\tau_k^m = G^m\left(-\frac{\langle \mathbf{w}, \mathbf{\Phi}(x_i,y_i)-\mathbf{\Phi}(x_k,y_k)\rangle}{\| \mathbf{\Phi}(x_i,y_i)-\mathbf{\Phi}(x_k,y_k)\|^2}\right) \quad (12)$$

$$G^m(u) = \begin{cases} 0 & u<0 \\ u & u\geq 0 \text{ and } y_k = z_i \\ u/(N-1) & u\geq 0 \text{ and } y_k \neq z_i \end{cases} \quad (13)$$

The MIRA algorithm has been applied to statistical machine translation in [11] and [12] and to parsing in [13].

3) Support Vector Machine (SVM): The SVM is also a linear classifier and its aim is to find a separating hyperplane that maximizes the margin between the nearest samples of two classes. The constrained optimization problem is defined as:

$$\min_{\mathbf{w}} \frac{1}{2}\langle \mathbf{w},\mathbf{w}\rangle + C\sum_j \xi_j$$
$$\text{subject to } c_j\langle \mathbf{w}, \mathbf{\Phi}_j(x,y)\rangle \geq 1-\xi_j \text{ and } \xi_j \geq 0 \quad (14)$$

where $c_j \in \{\pm 1\}$ are the class labels and ξ_j are the slack variables for violations of the margin constraints for the linearly nonseparable case. Note that here, the index j is not constrained within an N-best list and covers the whole sample set. C is a user-defined trade-off parameter between violations and smoothness. It is possible to assign different C values to the positive and negative classes, especially when the classes are not balanced: $C_+ = \beta C_-$. The major advantage of SVM is that this is a convex optimization problem that can be solved analytically unlike the perceptron that uses gradient-descent and risks getting stuck in local optima.

The labeling of training examples in an SVM setup is not straightforward. One can divide the hypotheses into positive and negative classes by setting a threshold either on the baseline recognition score, or the WER. In our implementation, we choose all hypotheses having the lowest error rate of their N-best list as the positive examples, and the rest as the negative examples.

SVMs have also been used as an alternative discrimination method. Using an English word n-gram setup, [14] applies the SVM classifier over an equal number of positive and (artificially generated) negative examples. It is reported that the accuracy of the system increases if negative sentences are disrupted more, and if the total number of examples is increased. Several modified versions of the SVM algorithm are also used for other language modeling tasks such as lexical disambiguation [15], parsing and machine translation [16].

C. Ranking Algorithms

Using classification, we define DLM as the separation of better examples from worse. But we know that the entries in the list have different degrees of goodness and hence it seems more natural to regard DLM as the reranking of the list where the number of word errors provides the target ranking for each possible transcription. This problem is similar to ordinal regression in that all examples $\mathbf{\Phi}(x,y)$ of the training set are given a rank r instead of a class label. However, unlike ordinal regression, in reranking the ranks are defined only between the examples of the same utterance, i.e., the N-best list.

In a reranking scenario, we would like to find $\mathbf{w}$ such that for any two hypotheses a and b from the same N-best list, if a has fewer word errors than b, it has a higher rank (is closer to the top of the list) than b. The model output differences should then be greater than some separation threshold $\lambda > 0$:

$$r_a \succ r_b \Longleftrightarrow \langle \mathbf{w}, \mathbf{\Phi}(x_a,y_a)-\mathbf{\Phi}(x_b,y_b)\rangle > \lambda \quad (15)$$

It must be noted that ranking ordering is the opposite of numeric ordering. For instance, if $r_a = 1$ and $r_b = 2$, then $r_a \succ r_b$.

1) Ranking Perceptron: The ranking perceptron algorithm integrates the reranking methodology into the perceptron setup [2]. Here the margin depends on the ranks and is proportional to the difference in ranks:

$$r_a \succ r_b \Longleftrightarrow \langle \mathbf{w}, \mathbf{\Phi}(x_a,y_a)-\mathbf{\Phi}(x_b,y_b)\rangle \geq \tau g(r_a,r_b) \quad (16)$$

where τ is a positive constant. We use the same $g(\cdot)$ function as in [2]:

$$g(r_a,r_b) = \begin{cases} \dfrac{1}{r_a}-\dfrac{1}{r_b} & r_a \succ r_b \\ 0 & r_a \prec r_b \end{cases} \quad (17)$$

This selection of the ranking function aims to achieve a greater separation between the hypotheses that are close to the top of the list than those at the bottom[1]. It also ensures that the following margin-rank relation is preserved:

$$r_a \succ r_b \succ r_c \Longleftrightarrow \begin{cases} g(r_a,r_c) > g(r_a,r_b) \\ g(r_a,r_c) > g(r_b,r_c) \end{cases} \quad (18)$$

Unlike [2], we use an additional learning rate parameter, η, and the update rule is:

$$\mathbf{w} = \mathbf{w} + \eta g(r_a,r_b)(\mathbf{\Phi}(x_a,y_a)-\mathbf{\Phi}(x_b,y_b)) \quad (19)$$

This learning rate is decreased by multiplying with a decay rate of $\gamma < 1$ at the end of each epoch and the weights are finally *averaged*, as done in Section II.B1.

Ranking and reranking adaptations of the perceptron algorithm have been around for some time: In [18], a perceptron ranking (PRanking) algorithm that divides the space into regions bounded by threshold levels is proposed. Each example is assigned a rank, and two neighboring ranks are set apart by a biased boundary. Note that this algorithm cannot be applied to DLM, as it needs global ranks, i.e., the ranks of hypotheses in different utterances need to be comparable with each other. Another study includes a review of reranking and introduces two perceptron-based ranking algorithms to decrease data complexity and training time [2]. Such algorithms have been applied to parse reranking and machine translation tasks in [19] and [20]. To the best of our knowledge, what we propose here, namely, ranking perceptron variants have not been applied to ASR output reranking before.

2) Ranking MIRA: We apply a modified ranking version of MIRA which updates the prototype if any pair of hypotheses with $r_a \succ r_b$ do not satisfy the margin requirement of $g(r_a,r_b)$.

$$\mathbf{w} = \mathbf{w} + \tau_{ab}(\mathbf{\Phi}(x_a,y_a)-\mathbf{\Phi}(x_b,y_b)) \quad (20)$$

[1]The effects of choosing another function in a different training setup is studied in [17].

$$\tau_{ab} = G^r\left(\frac{g(r_a, r_b) - \langle \mathbf{w}, \mathbf{\Phi}(x_a, y_a) - \mathbf{\Phi}(x_b, y_b)\rangle}{\| \mathbf{\Phi}(x_a, y_a) - \mathbf{\Phi}(x_b, y_b) \|^2}\right) \tag{21}$$

$$G^r(u) = \begin{cases} 0 & u < 0 \\ u & 0 \le u \le g(r_a, r_b) \\ g(r_a, r_b) & g(r_a, r_b) < u \end{cases} \tag{22}$$

3) Ranking SVM: The ranking SVM algorithm is a modification of the classical SVM setup to handle the reranking problem, defined as

$$\min_{\mathbf{w}} \frac{1}{2}\langle \mathbf{w}, \mathbf{w}\rangle + C\sum_{a,b}\xi_{ab}$$
$$\text{s.t. } \langle \mathbf{w}, \mathbf{\Phi}(x_a, y_a) - \mathbf{\Phi}(x_b, y_b)\rangle \ge 1 - \xi_{ab}$$
$$\forall (a,b) \in P, \quad \xi_{ab} > 0 \tag{23}$$

Here, C is again the trade-off value and P is the set of (a, b) pairs for which $r_a \succ r_b$. The constraint in (23) implies that the ranking optimization can also be viewed as an SVM classification problem on pairwise difference vectors, $\mathbf{\Phi}(x_a, y_a) - \mathbf{\Phi}(x_b, y_b)$. In that sense, the algorithm tries to find a large margin linear function which minimizes the number of pairs of training examples that need to be swapped to achieve the desired ranking [21].

The relationship between the ranking perceptron and ranking SVM algorithms is given in the Appendix.

In [22] the ranking SVM algorithm is compared with three other discriminative algorithms, namely perceptron, boosting, and minimum sample risk (an algorithm which applies line search to minimize the total error over a useful subset of features), in terms of accuracy and training time. Because of the long training time, the authors had to use 20-best lists instead of 100-best for their ranking SVM implementation. Despite this limitation, they show that this algorithm generalizes better, based on its performance on an unseen test set.

D. Data Sampling

The ranking algorithms include examples in a pairwise manner and complexity increases fast as the sample size and the number of unique ranks are increased. Our aim in this section is to investigate data sampling to relax some of these constraints and to decrease the complexity of the algorithm. We propose three approaches that work on the N-best lists, sorted first with respect to the number of word errors (WE) in ascending order, and then if WE are equal, with respect to the recognition scores in descending order. For the first and second approaches, the rank is taken as $1 + \text{WE}$:

- In *Uniform Sampling* (US), we select $n = \{2, 3, 5, 9\}$ instances in uniform intervals from this ordered list. For instance in US-5 with a 50-best list, the hypotheses with the indices 1, 13, 25, 37 and 50 are used only (The best and the worst hypotheses are always in the shortlist).
- *Rank Grouping* (RG) groups the hypotheses having the same unique WE and selects one or two examples as representatives from each group. In RG-1, this representative is the one having the highest score. In RG-2, we add the example with the lowest score.

TABLE I
AN EXAMPLE OF SAMPLING SCHEMES

Order	WE	US-2	US-3	US-5	RG-1	RC-2×3
1	0	1	1	1	1	1
2	1				2	1
3	2			3	3	1
4	2					
5	2		3	3		
6	3				4	
7	3			4		2
8	4				5	2
9	4	5	5	5		2

- In *Rank Clustering* (RC), we generate artificial rank clusters. We try RC-2 × 3 and RC-2 × 10, where we select 3 and 10 examples, respectively, from the top and bottom of the ordered list. Positive integers ($\{1, 2\}$) are assigned to these clusters as their ranks.

A simplified example of these sampling schemes is presented in Table I. The first column denotes the ordered hypotheses, with their WE shown in the second column. In the columns that follow, the rank values associated with these hypotheses are given.

The aim of uniform selection is to decrease directly the number of instances. It should be noted that each US scheme with increasing n also contains the instances from the previous ones. In RG, we would like to keep at least one instance from all available ranks, whereas in RC, we guarantee decreasing both the number of instances and ranks. Note that the sampling strategy we use here considers only single hypotheses but not hypothesis pairs.

Similar hypothesis selection schemes by sampling from an N-best list are investigated in [23] where it is argued that the selection of hypotheses based on WER should be preferred over the recognition score. Using the perceptron algorithm, the authors achieve an accurate and compact corrective model. In [20], a perceptron-like algorithm has been proposed that splits the training data into top r-ranked and bottom k-ranked translations. The same study also includes an ordinal regression algorithm for reranking in an machine translation task and improvements in BLEU of up to 0.3% are reported.

III. EXPERIMENTS

A. Experimental Setup

In this study, discriminative language modeling techniques are used in a Turkish broadcast news transcription task. We use a Turkish broadcast news data set constituted of approximately 194 hours of speech recorded from news channels. The data are divided into disjoint training (188 hours), held-out (3.1 hours) and test (3.3 hours) subsets, having 105 355, 1 947 and 1 784 utterances, respectively. The held-out and test sets contain about 23k words and 32k morphs. The acoustic model and the baseline ASR system were developed with the AT&T tools[2] and the generative language models were built using the SRILM toolkit[3]. These setups are the same as mentioned in [8], where a decrease

[2]http://www2.research.att.com/~fsmtools{fsm,dcd}/

[3]http://www.speech.sri.com/projects/srilm/

TABLE II
PERCEPTRON HELD-OUT WER (%)
(THE BEST RESULT IS SHOWN IN BOLDFACE)

w_0	0.0	1.0	2.0	4.0	8.0	16.0
WER	27.09	**22.14**	22.26	22.38	22.39	22.47

in word error rate of up to 0.8% has been reported with the averaged perceptron algorithm using basic sub-lexical features for the same task.

We use a statistical morph based ASR setup, as morphs obtained via the Morfessor algorithm[4] were shown to be more suitable and effective for the agglutinative nature of Turkish [24]. We use unigram, bigram and trigram morph models. There exist a total of 45 889 unique morphs, 2 272 714 morph pairs, and 9 242 126 morph triples in the dataset.

The hypotheses extracted from the ASR lattice are output as 50-best lists. The first element of the feature vector, $\Phi_0(x, y)$, is the log-probability of x in the lattice obtained from the baseline recognizer, and includes the contribution of baseline acoustic and generative language models. The rest of the feature vector consists of counts showing the number of times a particular morph n-gram occurs in the hypothesis. With such a large number of unique features, we have highly sparse feature vectors.

Using this setup, the generative baseline word error rates on the held-out and test set are 22.93% and 22.37%, respectively. If we were to select the oracle hypothesis for each utterance, the best rates that could be obtained from the same N-best list would be 14.18% and 13.92%, respectively.

B. Classification Algorithms

1) Perceptron: The perceptron (Per) algorithm (Section II.B1) with the 50-best morph unigram setup is the first of our discriminative training experiments. Unlike [1] and [8], to make sure that the perceptron considers the same set of candidate hypotheses for processing as the ranking variants, here we update $\mathbf{w}$ only if the WE of the current best is not equal to that of the oracle. The weight w_0 associated with Φ_0 has a different range than the rest and empirical results have shown that it is better to keep it fixed (unchanged) during perceptron training. Table II presents the held-out word error rates for different values of weight w_0, over at most three epochs over the data. Here, and in the tables that follow, the best result is shown in boldface.

For the optimal choice of w_0 and the number of epochs, the error rate on the held-out set still gives 22.14% as in [8], implying a reduction of 0.8% over the generative baseline. The same model yields 21.84% on the test data. An overall analysis on the test data and statistical significance information is given in Section IV.

2) MIRA: We experiment both with the single and multiple update MIRA variants. Similar to the perceptron setting, we keep w_0 fixed. We use Φ_0 in evaluating the score of the hypothesis with the models but we do not use it in the norm calculations. Table III represents the best WERs obtained for single and multiple updates for different values of w_0.

[4]http://www.cis.hut.fi/projects/morpho/

TABLE III
(S)INGLE AND (M)ULTIPLE MIRA HELD-OUT WER (%)

w_0	0.0	1.0	2.0	4.0	8.0	16.0
WER(S)	22.93	22.26	22.26	22.26	22.26	22.26
WER(M)	22.93	22.24	22.24	22.24	22.24	22.24

TABLE IV
CLASSIFICATION SVM HELD-OUT WER (%)

	$C = 1$	$C = 10$	$C = 100$
$\beta = 0.01$	22.85	22.85	22.85
$\beta = 0.1$	22.78	22.78	22.78
$\beta = 1$	**22.50**	22.81	22.52
$\beta = 10$	24.80	24.91	24.90
$\beta = 100$	29.28	29.63	29.26

TABLE V
RANKING PERCEPTRON HELD-OUT WER (%)

Updates	w_0 fixed	w_0 free
Online	**21.75**	21.84
Batch	21.80	21.81

We see that if w_0 is used (it is not zero), its value does not influence the result both in single and multiple update variants. We also see no significant difference between the accuracies of single and multiple MIRA and from now on, we report only single update results because it is simpler. The test set WER for this case is 21.83%.

3) SVM: Two-class SVM tests are evaluated using the LIBLINEAR toolbox [5]. The sample labeling method explained in Section II.B3 assigns approximately 10% of the examples to the positive class. Table IV shows the held-out set WERs with respect to some combinations of the trade-off parameter (C) and minority (positive) class weight (β).

With the optimal selection of parameters (penalty value and its weight across two classes), the lowest WER that could be obtained on the held-out set is 22.50%, which is better than the baseline but worse than the perceptron's and the MIRA's best. On the test set, the WER is calculated as 22.08%.

C. Ranking Algorithms

1) Ranking Perceptron: We have several factors to optimize during the training of the ranking perceptron (PerRank). The weight updates can be made in an online manner at each new instance (sample level), or in batch (cluster level) [2]. Furthermore, we have the option to choose whether to fix the weight w_0, just like in the perceptron, or update it with the other weights. Finally, optimal values of the algorithm parameters, η, γ, and τ must be searched for.

Table V shows the held-out WERs for different setup combinations with respect to the optimal choices of algorithm parameters and (at most) 20 epochs. Comparing the results with the ones in Table II, we see that the ranking perceptron performs better than the perceptron, and that the online update strategy with the w_0 parameter fixed must be preferred. The lowest WER of 21.75% is obtained using the parameter set $w_0 = 14.0, \eta = 1, \gamma = 0.9, \tau = 64$. Fig. 1 shows the change in WER with respect to the number of epochs for this setup. The test set error rate for the same experiment is 21.47%.

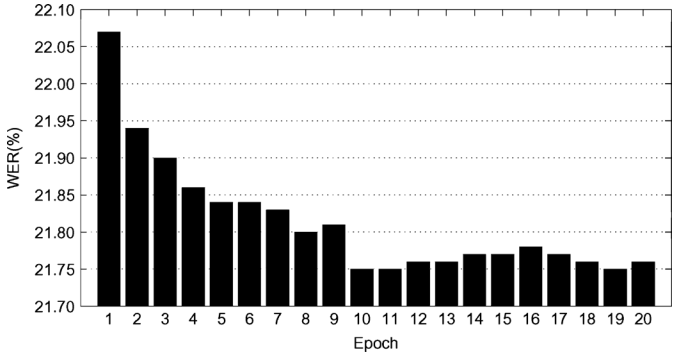

Fig. 1. PerRank held-out WER (%) with respect to the number of epochs.

TABLE VI
RANKING MIRA HELD-OUT WER (%)

w_0	0.0	1.0	2.0	4.0	8.0	16.0
WER	28.08	**22.16**	22.19	22.19	22.19	22.20

TABLE VII
RANKING SVM HELD-OUT WER (%)

C	1	100	1000	10000	20000
WER	22.34	22.26	22.13	22.14	**22.11**

2) Ranking MIRA: The held-out ranking MIRA (MIRArank) results shown in Table VI are close to those of the perceptron. Just like MIRA, the value of w_0 (unless zero) does not have a drastic influence. The test set error rate for the best case is 21.74%.

3) Ranking SVM: For the ranking SVM experiments, we use the $\mathrm{SVM}^{\mathrm{rank}}$ tool[5] which provides a fast implementation of the algorithm. Table VII shows the WERs obtained from the held-out set for different trade-off (C) values. The results suggest that ranking SVM leads to better results than the two-class SVM and is able to yield comparable results to those of the perceptron. Furthermore, on the test set it gives 21.61%, an additional improvement of 0.2%, which indicates that the model learned from the reranker generalizes better.

D. Increasing the Number of Features

Up to now we did experiments using a morph unigram (1g) setup, and obtained a lowest WER of 21.75% using the ranking perceptron. In this section we try to see how these six algorithms behave if we extend the feature space by adding higher order n-grams. Fig. 2 shows the held-out WERs of these experiments, all optimized within their parameter space.

As we can see from the figure, for all algorithms, adding bigram (2g) and trigram (3g) features does not decrease but increase the error rates, most possibly due to the lack of sufficient training data which leads to overfitting. This finding is coherent with the observations of [8]. Note that even in this case, the ranking perceptron algorithm shows superior performance to the other five.

E. Dimensionality Reduction

We know that reducing the number of dimensions eases the curse of dimensionality and drastically decreases space and time complexity. It also provides a better generalization by eliminating the effects of noise in redundant features. In this section, we go the other way around and apply feature selection[6]. In this study, we count the number of times each specific feature (morph n-gram) occurs in the training set, and set a threshold, below which that feature will be discarded. We call this approach Count-Based Thresholding (CBT).

[5]http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

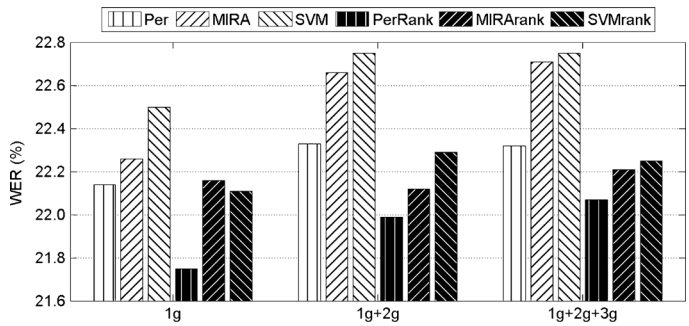

Fig. 2. Higher-order n-grams held-out WER (%).

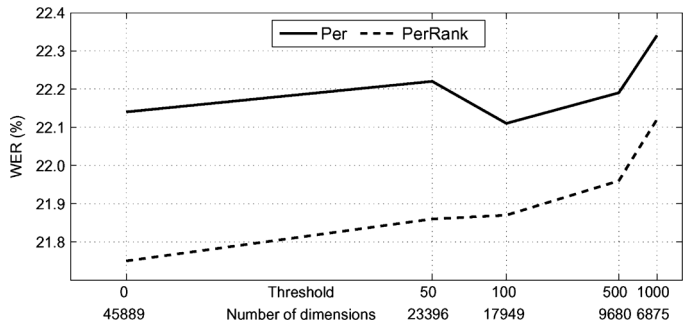

Fig. 3. Count-based thresholding held-out WER (%) for different thresholds and dimensions.

We tried this with the perceptron and ranking perceptron. Fig. 3 shows the 50-best morph unigram held-out set results for different values of the threshold, along with the number of dimensions in the reduced space. We see that the performance of Per is not degraded drastically even if we reduce the number of dimensions by one fifth, with the threshold of 500. On the other hand, a slight increase in WER is observed in PerRank, which can be explained by the utilized number of features by the algorithm. This finding will be explained in Section IV.B. Note that CBT results follow a similar trend for the bigram setup as well.

F. Sampling From the N-Best List

In Table VIII, we present held-out WERs of our three data sampling approaches in Section II.D for six algorithms, with an optimal parameter selection that yields the highest accuracy on the same set. Results of the 50-best unigram setup are also repeated for comparison.

The first notable result here is the 22.10% WER by the perceptron, which suggests that using only two examples (the best and worst in terms of WE) is as good as using all the examples in the list. This finding corroborates the result presented in [23]. Adding more examples to the training set does not decrease the error rate further with this method. SVM, being the

[6]To decrease the dimensionality, we also tried feature extraction: In our earlier study [1], we tried applying online PCA, but did not obtain any considerable gain due most probably to the restriction to a linear feature extractor.

TABLE VIII
SAMPLING SCHEMES HELD-OUT WER (%)

Method	Per	MIRA	SVM	PerRank	MIRArank	SVMrank
All (50)	22.14	**22.26**	22.50	**21.75**	22.16	22.11
US-2	**22.10**	22.68	22.36	21.99	22.62	22.66
US-3	22.14	22.51	22.46	21.94	22.28	22.21
US-5	22.22	22.58	22.58	21.83	22.12	**21.86**
US-9	22.18	22.37	22.59	21.77	**21.97**	21.89
RG-1	22.26	22.41	**22.29**	21.97	22.07	22.11
RG-2	22.27	22.45	22.49	21.90	22.01	23.04
RC-2×3	22.29	22.58	22.40	21.93	22.28	22.39
RC-2×10	22.20	22.40	22.48	21.81	22.12	22.10

weakest method in 50-best, prefers a sampled training set but cannot compete with Per.

MIRA performs better with more hypotheses. Though it is similar to Per, it is not as accurate, neither in classification nor ranking.

Unlike Per, SVMrank benefits from increasing the number of examples in the US and RC cases, but not in RG. The best result obtained here is 21.86% with the US-5 sampling scheme. This value is better than using 5-best lists (22.11%) or choosing 5 examples randomly (22.24%). The RG and RC schemes also provide comparable results to the baseline.

The superiority of the ranking perceptron is once more evident in Table VIII. The algorithm outperforms others in all schemes and responds positively to increasing the sample size. As in SVMrank, the accuracy improves as the number of selected hypotheses increases.

IV. DISCUSSION

So far we have only compared the models by looking at their held-out set accuracies. However the learning process reveals some other important side information about the performance and working behavior of these methods. In this section, we compare the algorithms with respect to their CPU times, model complexities, test set accuracies, and we check for differences that are statistically significant. In the following sections, we use the 50-best unigram setup in the experiments.

A. Comparison of CPU Times

We denoted earlier that ranking algorithms have higher complexity than classification algorithms, and that they need more computational resources as the number of instances and unique ranks are increased. Training the models by processing all the hypotheses takes <2 mins with Per, <3 mins with MIRA and <5 mins with SVM, whereas it takes <30 mins with PerRank and MIRArank.

In terms of running time, SVMrank is much more costly than the other models. Even though $\mathrm{SVM}^{\mathrm{rank}}$ toolkit provides a fast and efficient implementation, training is much slower. In Table IX, the total number of training instances and the elapsed CPU times for the SVMrank setup are shown with respect to different data sampling methods. This time a fixed trade-off value of $C = 1000$ is used for fair comparison. We see that although SVMrank training takes time in the ranges of hours, by an intelligent sampling technique from the N-best list, it is possible to decrease the number of training instances and thus the CPU times considerably, while keeping the WERs still in a tolerable region.

TABLE IX
TRAINING CPU TIMES OF SVMRANK FOR FIXED $C = 1000$

Method	Number of instances	CPU hours	Held-out WER(%)
All	4,939,368	25:00	22.26
US-2	209,675	0:14	22.77
US-3	313,120	0:51	22.50
US-5	518,234	1:42	**22.05**
US-9	923,770	3:34	22.20
RG-1	466,277	1:49	**22.01**
RG-2	867,045	4:26	23.14
RC-2×3	620,160	3:16	22.59
RC-2×10	2,020,450	16:45	22.21

B. Comparison of Models

We also compare the performances of six models in terms of their complexity and accuracy. Since all the models are linear, simpler models are the ones which use fewer features. The weights of unused features are zero at the end of the training and the complexities of the models are compared in terms of the features they use and they share; we consider weights not exceeding the threshold 10^{-4} as zero. In Table X(a)–(d), we compare models with respect to the number of zero and nonzero features they use.

Though the perceptron and MIRA use fewer than half of the features, the other models use almost all of them, due to the fact that the latter consider most of the hypotheses in their updates as opposed to only two (the oracle and the current best). It should also be noted that though two models might use the same features, their weights can have different signs.

Feature comparisons of perceptron and MIRA variants for 2g and 3g datasets are given in Table XI(a) and (b) (SVM and SVMrank results cannot be obtained here due to their infeasible training times). Similar to the unigram case, ranking algorithms use many more features.

C. Comparison of Test Set Accuracies

The models are also statistically compared in terms of their accuracy on the test set. WERs shown in Table XII reveal some other important information. The ranking perceptron generalizes better than all the other methods except the ranking SVM. As seen in Table XIII, the test error improvements over Per is significant at $p = 0.003$ for PerRank, as measured by the NIST MAPSSWE test[7]. The test shows that there is no statistically significant difference between PerRank and SVMrank, nor between SVMrank and Per. But SVM has higher WER than both PerRank ($p < 0.001$) and SVMrank ($p = 0.004$). MIRA does not differ from the other methods except PerRank ($p = 0.002$). MIRArank differs from SVM and PerRank. The ordering found is

PerRank SVMrank MIRArank MIRA Per SVM

D. Comparison of Statistical Significance

Results in the previous section are over a single run. To average over randomness, we also apply 10-fold cross validation by splitting the training dataset into 10-partitions, and results on the test set are given in Table XIV. For each model, we use the

[7]http://www.itl.nist.gov/iad/mig/tools/

TABLE X
PAIRWISE COMPARISON OF MODELS IN TERMS OF THE NUMBER OF ZERO (Z) AND NONZERO (NZ) FEATURES THEY USE ON UNIGRAMS. (a) PERCEPTRON AND SVM, (b) PERCEPTRON AND MIRA, (c) SVM AND MIRA, (d) SVMRANK AND MIRARANK

(a)

		Per		SVMrank	
		NZ	Z	NZ	Z
SVM	NZ	19,946	25,223	44,390	779
	Z	175	545	710	10
PerRank	NZ	20,119	24,896	44,291	724
	Z	2	872	809	65

(b)

		MIRA		PerRank	
		NZ	Z	NZ	Z
MIRArank	NZ	20,540	23,290	43,767	63
	Z	7	2,052	1,248	811
Per	NZ	18,728	1,393		
	Z	1,819	23,949		

(c)

		MIRA	
		NZ	Z
SVM	NZ	20,481	24,688
	Z	66	654

(d)

		MIRArank	
		NZ	Z
SVMrank	NZ	43,790	1,310
	Z	40	749

TABLE XI
FEATURE COMPARISON FOR PERCEPTRON AND MIRA. (a) BIGRAMS, (b) TRIGRAMS

(a)

		PerRank		MIRA	
		NZ	Z	NZ	Z
Per	NZ	257,458	44	203,719	53,783
	Z	1,924,116	91,096	44,954	1,970,258
MIRArank	NZ	2,079,702	7,039	248,541	1,838,200
	Z	101,872	84,101	132	185,841

(b)

		PerRank		MIRA	
		NZ	Z	NZ	Z
Per	NZ	819,233	520	703,725	116,028
	Z	7,891,409	430,964	127,667	8,194,706
MIRArank	NZ	8,023,671	45,944	830,101	7,239,514
	Z	686,971	385,540	1,291	1,071,220

TABLE XII
TEST SET WER (%)

Per	MIRA	SVM	PerRank	MIRArank	SVMrank
21.84	21.83	22.08	21.47	21.74	21.61

parameters which yield the minimum WER over the held-out set. We apply 10-fold cross-validation paired t test on the test WERs and Table XV shows the p-values obtained by pairwise t tests. This time all the differences are significant and we have a clear ordering:

$$\text{PerRank} < \text{SVMrank} < \text{MIRArank} < \text{Per} < \text{MIRA} < \text{SVM}$$

TABLE XIII
RESULTS (P VALUES) OF MAPSSWE TEST

	MIRArank	PerRank	Per	SVMrank	SVM
MIRA	0.638	0.002	0.795	0.121	0.156
SVM	0.014	< 0.001	0.085	0.004	
SVMrank	0.168	0.226	0.131		
Per	0.764	0.003			
PerRank	0.007				

V. CONCLUSION

Being an agglutinative language with rich morphological structure, Turkish is a challenging language in terms of obtaining large vocabulary continuous speech recognition (LVCSR) accuracy. The possibility of producing almost infinitely many words from a single stem results in very high WERs, as compared to systems of similar vocabulary sizes in English. It has been shown in previous studies that using sub-lexical units such as statistical morphs instead of word-based models, help enhance the recognition performance [25].

In this paper, we present some directions to enhance discriminative language modeling performance for Turkish broadcast news transcription. We use and compare three algorithms, namely, perceptron, MIRA and SVM, both for classification and ranking. We apply thresholding as a dimensionality reduction technique on the sparse feature set, and some sample selection strategies to decrease the complexity of training.

We see first that ranking leads to lower WER than classification; this is true for all algorithms. The disadvantage is that ranking takes more time and this is where feature selection and sampling becomes more interesting.

The use of SVM leads to no significant decrease in error and may even worsen performance. SVM also takes longer to train. The complexity of SVM training can also be curbed significantly (from a day to minutes) by intelligent sampling from the N-best list. We note however that these are SVM results using the linear kernel, and with better, application-specific kernels, SVM may provide more interesting results.

SVMrank uses constant penalty whereas perceptron ranker uses the $g(r_a, r_b)$ function and that is how SVM ranker may be improved; see the Appendix for a more detailed comparison between the two rankers.

Ranking algorithms, though more accurate, use more features than classification algorithms. The reason is that they do more updates while the classification algorithms make only one update for each hypothesis set. Another downside is that the ranking algorithms take more time to train due to more updates.

MIRA variants do not show significant improvements compared to their perceptron correspondents. A possible reason might be that the normalizing effect of the norms causes smaller updates diminishing their correcting effects. Another point is that as long as Φ_0 is used (w_0 is not zero), the other coefficients can adapt themselves accordingly and the same accuracy can be attained.

Using higher order n-gram data does not improve the performance but it increases the complexity and the training time. It is possible to do feature selection and use a subset of the unigram

TABLE XIV
10-FOLD TEST SET WER (%)

Per	MIRA	SVM	PerRank	SVMrank	MIRArank
21.87 ± 0.06	21.93 ± 0.05	22.19 ± 0.12	21.47 ± 0.05	21.63 ± 0.09	21.81 ± 0.03

TABLE XV
RESULTS (P VALUES) OF 10-FOLD CROSS-VALIDATION T TESTS

	MIRArank	PerRank	Per	SVMrank	SVM
MIRA	2.2×10^{-5}	6×10^{-9}	0.0418	2.4×10^{-5}	3.2×10^{-4}
SVM	9.3×10^{-6}	1.8×10^{-8}	2×10^{-5}	4×10^{-7}	
SVMrank	7.8×10^{-4}	2.1×10^{-3}	1×10^{-4}		
Per	0.9×10^{-4}	3×10^{-8}			
PerRank	1.9×10^{-9}				

features thereby decreasing complexity without losing from accuracy. The unigram representation is very sparse and high-dimensional and an interesting future research direction is to represent such a long sparse vector using fewer features.

We are currently investigating the effect of using larger N-best and feature sets. Our future directions include trying alternative definitions of the margin function, sampling from the hypothesis cross-product space, using the perceptron and ranking SVM for feature selection and data summarization, together with (possibly nonlinear) dimensionality reduction methods while hopefully preserving sparsity.

APPENDIX
RELATIONSHIP BETWEEN RANKING PERCEPTRON AND RANKING SVM

If $r_a \succ r_b$, we require $f(a) > f(b)$ where $f(u) = \langle \mathbf{w}, \mathbf{\Phi}_u \rangle$. With the ranking perceptron, the update rule is

$$\mathbf{w} = \mathbf{w} + g(r_a, r_b)(\mathbf{\Phi}_a - \mathbf{\Phi}_b) \tag{24}$$

This is applied when $f(a) < f(b)$. So the error function we should minimize is

$$E = \sum_{r_a \succ r_b, f(a) < f(b)} [f(b) - f(a)] \tag{25}$$

If we use gradient-descent, we get

$$\Delta \mathbf{w} = -\eta[\nabla_{\mathbf{w}} f(b) - \nabla_{\mathbf{w}} f(a)] = \eta(\mathbf{\Phi}_a - \mathbf{\Phi}_b)$$

If we penalize differences with respect to their rank differences as given by a function such as $g(r_a, r_b)$:

$$E = \sum_{r_a \succ r_b, f(a) < f(b)} [f(b) - f(a)] g(r_a, r_b) \tag{26}$$

when we use gradient-descent, we get

$$\Delta \mathbf{w} = \eta g(r_a, r_b)(\mathbf{\Phi}_a - \mathbf{\Phi}_b)$$

which is the update rule of (24) with $\eta = 1$.

Let us now consider the case of SVM. We require $f(a) > f(b)$, so when this is not satisfied, we need slack variables ξ_{ab} to make up for the difference:

$$f(b) \leq f(a) + \xi_{ab}, \quad \forall r_a \succ r_b$$

and the total error is

$$\begin{aligned} \min \quad & \sum_{r_a \succ r_b} \xi_{ab} \\ \text{subject to} \quad & f(a) \geq f(b) - \xi_{ab} \end{aligned} \tag{27}$$

If we require a difference of at least 1 unit as the margin, the constraints become

$$\text{subject to} \;\; f(a) \geq f(b) + 1 - \xi_{ab}$$

We can add a L_2 regularizer for smoothness and the error becomes

$$\min \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle + C \sum_{r_a \succ r_b} \xi_{ab}$$

where C denotes the relative weights of the first regularizer and the second data-misfit terms. With the perceptron too, if we like we can add a similar term—this is known as "weight decay" in neural network terminology.

So we see that, as would be expected, the ranking perceptron and ranking SVM minimize very similar error functions with some slight differences: (1) SVM enforces a minimum margin between differences, (2) In perceptron, error terms are weighted by the $g(r_a, r_b)$ term whereas for SVM, all have the same weight of C, and (3) SVM has an additional regularizer term—in perceptron, we have tried a version with weight decay but this did not a cause a significant difference.

ACKNOWLEDGMENT

The authors would like to thank Ebru Arısoy for the baseline DLM setup.

REFERENCES

[1] E. Dikici, M. Semerci, M. Saraçlar, and E. Alpaydın, "Data sampling and dimensionality reduction approaches for reranking ASR outputs using discriminative language models," in *Proc. Interspeech*, 2011, pp. 1461–1464.
[2] L. Shen and A. K. Joshi, "Ranking and reranking with perceptron," *Mach. Learn.*, vol. 60, pp. 73–96, Sep. 2005.
[3] B. Roark, M. Saraçlar, and M. Collins, "Discriminative n-gram language modeling," *Comput. Speech Lang.*, vol. 21, no. 2, pp. 373–392, Apr. 2007.
[4] K. Crammer and Y. Singer, "Ultraconservative online algorithms for multiclass problems," *J. Mach. Learn. Res.*, vol. 3, pp. 951–991, 2003.
[5] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, 2008.
[6] M. Collins, "Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms," in *Proc. EMNLP*, 2002, pp. 1–8.
[7] E. Arısoy, B. Roark, I. Shafran, and M. Saraçlar, "Discriminative n-gram language modeling for Turkish," in *Proc. Interspeech*, 2008, pp. 825–828.
[8] E. Arısoy, M. Saraçlar, B. Roark, and I. Shafran, "Discriminative language modeling with linguistic and statistically derived features," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 20, no. 2, pp. 540–550, Feb. 2012.

[9] Z. Li and S. Khudanpur, "Large-scale discriminative n-gram language models for statistical machine translation," in *Proc. 8th AMTA Conf.*, Oct. 2008, pp. 133–142.
[10] P. Jyothi and E. Fosler-Lussier, "Discriminative language modeling using simulated ASR errors," in *Proc. Interspeech*, 2010, pp. 1049–1052.
[11] T. Watanabe, J. Suzuki, H. Tsukada, and H. Isozaki, "Online large-margin training for statistical machine translation," in *Proc. EMNLP-CoNLL*, Jun. 2007, pp. 764–773.
[12] D. Chiang, Y. Marton, and P. Resnik, "Online large-margin training of syntactic and structural translation features," in *Proc. EMNLP*, 2008, pp. 224–233.
[13] R. McDonald, K. Crammer, and F. Pereira, "Online large-margin training of dependency parsers," in *Proc. ACL*, 2005, pp. 91–98.
[14] M. Vuorinen, M. Kaariainen, J. Rousu, Z. Wang, P. Mahe, and C. Cancedda, "D3.2 Learning methods for discriminative language models," SMART Project Rep., Jun. 2008.
[15] S. Bergsma, D. Lin, and D. Schuurmans, "Improved natural language learning via variance-regularization support vector machines," in *Proc. CoNLL*, 2010, pp. 172–181.
[16] C. Cherry and C. Quirk, "Discriminative, syntactic language modeling through latent SVMs," in *Proc. 8th AMTA Conf.*, Oct. 2008, pp. 65–74.
[17] E. Dikici, A. Celebi, and M. Saraçlar, "Performance comparison of training algorithms for semi-supervised discriminative language modeling," in *Proc. Interspeech*, 2012.
[18] K. Crammer and Y. Singer, "Pranking with ranking," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2001, vol. 14, pp. 641–647.
[19] L. Shen and A. K. Joshi, "Flexible margin selection for reranking with full pairwise samples," in *Proc. 1st Int. Joint Conf. Natural Lang. Process. (IJCNLP)*, 2004, pp. 446–455.
[20] L. Shen, A. Sarkar, and F. Och, "Discriminative reranking for machine translation," in *Proc. HLT-NAACL*, 2004, pp. 177–184.
[21] T. Joachims, "Optimizing search engines using clickthrough data," in *Proc. ACM SIGKDD Conf. Knowl. Disc. Data Mining (KDD)*, 2002, pp. 133–142.
[22] Z. Zhou, J. Gao, F. Soong, and H. Meng, "A comparative study of discriminative methods for reranking LVCSR n-best hypotheses in domain adaptation and generalization," in *Proc. ICASSP*, 2006, pp. 141–144.
[23] T. Oba, T. Hori, and A. Nakamura, "An approach to efficient generation of high-accuracy and compact error-corrective models for speech recognition," in *Proc. Interspeech*, 2007, pp. 1753–1756.
[24] E. Arısoy, M. Saraçlar, B. Roark, and I. Shafran, "Syntactic and sub-lexical features for Turkish discriminative language models," in *Proc. ICASSP*, 2010, pp. 5538–5541.
[25] E. Arısoy, D. Can, S. Parlak, H. Sak, and M. Saraçlar, "Turkish broadcast news transcription and retrieval," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 17, no. 5, pp. 874–883, Jul. 2009.

Erinç Dikici (S'08) received the M.S. degree from the Electrical and Electronics Engineering Department at Boğaziçi University, Istanbul, Turkey in 2009 and the B.S. degree from the Electronics and Communication Engineering Department at Istanbul Technical University in 2006. He is currently a Ph.D. student and a research assistant at the Speech Processing Group of BUSIM—Boğaziçi University Center for Signal and Image Processing. His dissertation is on supervised and semi-supervised methods for discriminative language modeling for Turkish.

Murat Semerci (S'12) received his B.S. degrees from the Electrical and Electronics Engineering Department and the Department of Computer Engineering (Double Major Program), Boğaziçi University, in 2005 and his M.S. degrees from the Department of Computer Engineering, Boğaziçi University, in 2007 and from the Computer Science Department, Rensselaer Polytechnic Institute, Troy, NY, in 2010. Currently he is a Ph.D. student in Boğaziçi University. His research interests include pattern recognition, kernel machines and machine learning.

Murat Saraçlar (M'00) received his B.S. degree in 1994 from the Electrical and Electronics Engineering Department at Bilkent University, M.S.E. degree in 1997 and Ph.D. degree in 2001 from the Electrical and Computer Engineering Department at the Johns Hopkins University. He is currently an associate professor at the Electrical and Electronic Engineering Department of Boğaziçi University. From 2000 to 2005, he was with the Multimedia Services Department at AT&T Labs Research. Dr. Saraçlar was a member of the IEEE Signal Processing Society Speech and Language Technical Committee (2007–2009). He is currently serving as an associate editor for IEEE SIGNAL PROCESSING LETTERS and IEEE IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING. He is on the editorial boards of *Computer Speech and Language*, and *Language Resources and Evaluation*.

Ethem Alpaydın (SM'04) received his B.Sc. from the Department of Computer Engineering of Boğaziçi University in 1987 and Ph.D. from École Polytechnique Fédérale de Lausanne in 1990. He has been teaching at the Department of Computer Engineering of Boğaziçi University since 1991, currently as a professor. His book Introduction to Machine Learning was published by The MIT Press in October 2004 and since has been translated to German, Chinese and Turkish.