An Online Relevant Set Algorithm for Statistical Machine Translation

* Christoph Tillmann and Tong Zhang

***Abstract*—This paper presents a novel online relevant set algorithm for a linearly-scored block sequence translation model. The key component is a new procedure to directly optimize the global scoring function used by a statistical machine translation (SMT) decoder. This training procedure treats the decoder as a black-box, and thus can be used to optimize any decoding scheme. The novel algorithm is evaluated using different feature types: 1) commonly used probabilistic features, such as translation, language, or distortion model probabilities, and 2) binary features. In particular, encouraging results on a standard Arabic-English translation task are presented for a translation system that uses only binary feature functions. To further demonstrate the effectiveness of the novel training algorithm, a detailed comparison with the widely used minimum-error-rate (MER) training algorithm [2] is presented using the same decoder and feature set. The online algorithm is simplified by introducing so-called 'seed' block sequences which enable the training to be carried out without a gold standard block translation. While the online training algorithm is extremely fast, it also improves translation scores over the MER algorithm in some experiments.**

***Index Terms*—Statistical machine translation, online algorithm, discriminative learning.**

EDICS Category: SLP-SSMT

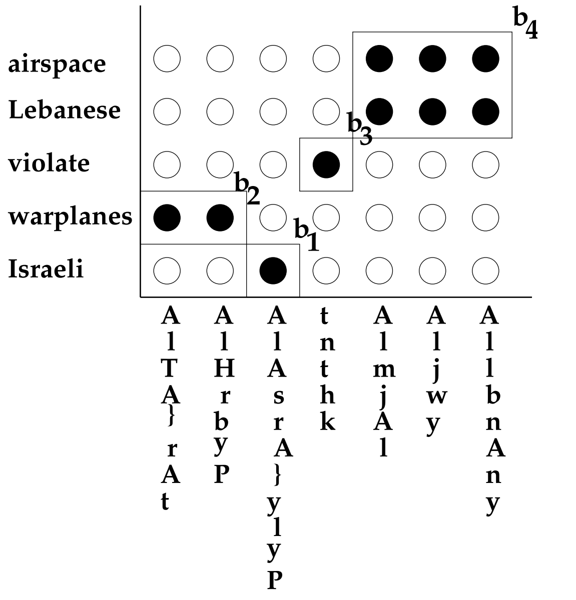

Fig. 1. An Arabic-English block sequence translation example, where the Arabic words are romanized. The following orientation sequence is generated: $o_1 = N, o_2 = L, o_3 = N, o_4 = R$.

I. Introduction

This paper employs a view of phrase-based SMT as a sequential process that generates block orientation sequences. A block is a pair of phrases which are translations of each other. For example, Figure 1 shows an Arabic-English translation example that uses four blocks. During decoding, we view translation as a block segmentation process, where the input sentence is segmented from left to right and the target sentence is generated from bottom to top, one block at a time. A block orientation sequence is generated under the restriction that the concatenated source phrases of all the blocks in the sequence yield the input sentence. This block sequence is monotone except for the possibility to swap neighbor blocks. In this local reordering model similar to [3], [4] a block b with orientation o is generated relative to its predecessor block b'. During decoding, we maximize the score $s_w(b_1^n, o_1^n)$ of a block orientation sequence (b_1^n, o_1^n):

$$s_w(b_1^n, o_1^n) = \sum_{i=1}^{n} w^T \cdot f(b_i, o_i, b_{i-1}), \tag{1}$$

where b_i is a block, b_{i-1} is its predecessor block, $o_i \in \{L(\text{eft}), R(\text{ight}), N(\text{eutral})\}$ is a three-valued orientation component linked to the block b_i, and n is the number of blocks. $f(b_i, o_i, b_{i-1}) \in \mathbb{R}^M$ is a high-dimensional feature representation of the block orientation pair (b_i, o_i, b_{i-1}), where M might be in the tens of millions. Most of the time the block b_i is generated immediately to the left ($o_i = L$) or the right ($o_i = R$) of its predecessor block b_{i-1}, where the orientation o_{i-1} of the predecessor block is ignored. The neutral orientation is used to handle blocks that are 'detached' from their predecessor blocks. This restricted reordering model results in fast decoding speeds while producing good translations results on the Arabic-English translation task.

The paper focuses on the discriminative training of the weight vector $w \in \mathbb{R}^M$ in Eq. 1. The decoding process is decomposed into local decision steps based on Eq. 1, but the model is trained in a global setting. The advantage of this approach is that it can easily handle tens of millions of features, e.g. up to 35 million features for the experiments in this paper. Moreover, under this view, SMT becomes quite similar to sequential natural language annotation problems such as part-of-speech tagging and shallow parsing, and the novel training algorithm in this paper is quite similar to work on training algorithms for these task, e.g. the online algorithm

* C. Tillmann is with the IBM T.J. Watson Research Center, Yorktown Heights, N.Y., 10598 USA, phone: 1-914-945-2705, fax: 1-914-945-4490, e-mail: ctill@us.ibm.com.

T. Zhang is with Yahoo! Research, New York City, N.Y. 10011 USA, phone: 1-646-351-5449, fax: 1-530-684-7183, e-mail: tongz@rci.rutgers.edu.

A preliminary version of this paper was published as [1].

for dependency parsing [5] and the perceptron algorithm for POS tagging [6]. The online relevant set algorithm is capable of achieving good translation results on a standard translation task without using specialized probability features as in [2]. Furthermore, a detailed comparison of the novel training algorithm with the widely-used minimum-error-rate (MER) training [2] is presented. Largely identical or even improved translation scores are obtained when comparing the MER training and the novel online algorithm using the same decoder and parameter setting.

The paper is structured as follows: Section II presents the baseline block sequence model and the feature representation. Section III presents the discriminative training algorithm that learns a good global ranking function used during decoding. In Section IV implementation details for the novel discriminative training algorithm are given. Section V introduces a perceptron-style simplification with competitive results when compared to the widely used MER training algorithm. Section VI shows results on a standard Arabic-English translation task for all the discriminative training algorithms presented in this paper. Finally, some discussion and future work are presented in Section VII.

II. Block Sequence Model

This paper views phrase-based SMT as a block sequence generation process. A block $b = (S, T)$ is a phrase pair consisting of source phrase S and target phrase T. Within the block sequence generation process local phrase reordering is handled explicitly by introducing block orientation as shown below. Starting point for the block-based translation model is a block set, e.g. about 9.5 million Arabic-English phrase pairs for the experiments in this paper. This block set is used to decode training sentence to obtain block orientation sequences that are used in the discriminative parameter training. Nothing but the block set and the parallel training data is used to carry out the training, i.e. while a translation and distortion model is used in generating the block set, these translation probabilities are not used during the discriminative training. During decoding, we maximize the score $s_w(b_1^n, o_1^n)$ of a block orientation sequence (b_1^n, o_1^n) as defined in Eq. 1. In modeling a block sequence, we emphasize adjacent block neighbors that have $o = R$ (Right) or $o = L$ (Left) orientation. As is demonstrated in Figure 1 during the bottom up decoding process each block b_i immediately follows block b_{i-1} in the target sentence, and its orientation label o_i depends on its source position relative to b_{i-1}. If the block b_i appears left adjacent to the source phrase of block b_{i-1} its orientation label is $o_i = L$. If the block b_i appears right adjacent to the source phrase of block b_{i-1} its orientation label is $o_i = R$. If the block b_i appears to the right of the source phrase of b_{i-1} but is not adjacent it is said to have neutral orientation $o_i = N$. Here, the source phrase of a single block must fill the source gap between successor block b_i and predecessor block b_{i-1}. This way blocks with neutral orientation are less strongly 'linked' to their predecessor block. If the initial block b_0 occurs at the sentence start it has right orientation $o_0 = R$. Since there is no predecessor block here, we assume the presence of a sentence 'boundary' block. Similar approaches to phrase-based SMT that restrict the word reordering to local phrase reordering only are presented in [3], [4], [7], [8].

Rather than predicting local block neighbors as in [3], here the model parameters w in Eq. 1 are trained in a global setting. Starting with a simple initial model, the training data is decoded multiple times: the weight vector w is trained to discriminate block sequences with a high translation score against block sequences with a high BLEU score. High scoring block sequences may contain translation errors that are quantified by a lower BLEU score. A 'true' high BLEU-scoring block sequence as well as the high scoring block sequences are represented by high dimensional feature vectors using the binary features defined below and the translation process is handled as a multi-class classification problem in which each block sequence represents a possible class. The effect of this training procedure can be seen in Figure 2 in Section VI: each decoding step on the training data adds a high-scoring block sequence to the discriminative training and the BLEU score on the training data is improved after each iteration (along with the test set BLEU score). A theoretical justification for this training procedure in terms of the novel relevant set algorithm is given in Section III. The algorithm can be adapted to work with different evaluation metrics as well, e.g. in Section VI-D the multi-reference word error rate (mWER) is used as training criterion. High BLEU scoring block sequences for each training sentence are obtained as follows: the regular phrase-based decoder is modified in a way that it uses the BLEU score as optimization criterion (independent of any translation model). Here, searching for the highest BLEU scoring block sequence is restricted to local block swapping as is the model-based decoding (as shown in Figure 1). The BLEU score is computed with respect to the single reference translation provided by the parallel training data. A block sequence with an average BLEU score of about 54.2 % is obtained for each training sentence. The training BLEU score is computed for each training sentence pair separately (treating each sentence pair as a single-sentence corpus with a single reference) and then averaged over all training sentences. Although block sequences are found with a high BLEU score on average there is no guarantee to find the maximum BLEU block sequence for a given sentence pair. The target word sequence corresponding to a block sequence does not have to match the reference translation, i.e. maximum BLEU scores are quite low for some training sentences. In order to make the comparison with the MER training [2] straightforward a simplification using so-called 'seed' block sequences is introduced in Section V. It enables the training to be carried out without an initial gold standard block sequence.

Feature vector components in the block bigram feature vector $\mathbf{f}(b_i, o_i, b_{i-1}) \in \mathbb{R}^M$ in Eq. 1 can be both real-valued and binary-valued. As real-valued feature vector components we typically take the negative logarithm of some block model probabilities (for details see Section VI-B). The binary feature functions are similar to feature functions used in common phrase-based translation systems. While the use of POS-based features does not improve translation performance in the current experiments, more sophisticated features will be tested in future experiments. For illustration purposes, the binary features do 'fire' on the example block sequence in Figure 1. There are **phrase-based** and **word-based** features:

$$f_1(b_i, o_i, b_{i-1}) = \begin{cases} 1 & \text{block } b_i \text{ consists of target phrase} \\ & \text{'violate' and source phrase 'tnthk'} \\ 0 & \text{otherwise} \end{cases}$$

$$f_2(b_i, o_i, b_{i-1}) = \begin{cases} 1 & \text{'Lebanese' is a word in the target} \\ & \text{phrase of block } b_i \text{ and 'AllbnAny'} \\ & \text{is a word in the source phrase} \\ 0 & \text{otherwise} \end{cases}$$

Example feature f_1 is a 'unigram' phrase-based feature capturing the identity of a block. Additional phrase-based features include block orientation, target and source phrase bigram features. Word-based features are used as well, e.g. example feature f_2 captures word-to-word translation dependencies similar to the use of Model 1 probabilities in [9]. Additionally, we use distortion features involving relative source word position and m-gram features for adjacent target words. These features correspond to the use of a language model, but their weights are trained on the parallel training data only. For the most complex model, the number of features is about 35 million (ignoring all features that occur only once).

III. Approximate Relevant Set Method

Throughout the section, we let $\mathbf{z} = (b_1^n, o_1^n)$. Each block sequence $\mathbf{z} = (b_1^n, o_1^n)$ corresponds to a candidate translation. In the training data where target translations are given, a BLEU score $\mathrm{Bl}(\mathbf{z})$ can be calculated for each $\mathbf{z} = (b_1^n, o_1^n)$ against the target translations. In this set up, our goal is to find a weight vector w such that the higher $s_w(\mathbf{z})$ is, the higher the corresponding BLEU score $\mathrm{Bl}(\mathbf{z})$ should be. If we can find such a weight vector, then block decoding by searching for the highest $s_w(\mathbf{z})$ will lead to good translation with high BLEU score. Formally, we denote a source sentence by $\mathbf{S}$, and let $V(\mathbf{S})$ be the set of possible candidate oriented block sequences $\mathbf{z} = (b_1^n, o_1^n)$ that the decoder can generate from $\mathbf{S}$. For example, in a monotone decoder, the set $V(\mathbf{S})$ contains block sequences $\{b_1^n\}$ that cover the source sentence $\mathbf{S}$ in the same order. For a decoder with local reordering, the candidate set $V(\mathbf{S})$ also includes additional block sequences with reordered block configurations that the decoder can efficiently search. Therefore depending on the specific implementation of the decoder, the set $V(\mathbf{S})$ can be different. In general, $V(\mathbf{S})$ is a subset of all possible oriented block sequences $\{(b_1^n, o_1^n)\}$ that are consistent with input sentence $\mathbf{S}$. Given a scoring function $s_w(\cdot)$ and an input sentence $\mathbf{S}$, we can assume that the decoder implements the following decoding rule:

$$\hat{\mathbf{z}}(\mathbf{S}) = \arg \max_{\mathbf{z} \in V(\mathbf{S})} s_w(\mathbf{z}). \tag{2}$$

Let $\mathbf{S}_1, \ldots, \mathbf{S}_N$ be a set of N training sentences. Each sentence $\mathbf{S}_i$ is associated with a set $V(\mathbf{S}_i)$ of possible translation block sequences that are search-able by the decoder. Each translation block sequence $\mathbf{z} \in V(\mathbf{S}_i)$ induces a translation, which is then assigned a BLEU score $\mathrm{Bl}(\mathbf{z})$ (obtained by comparing against the target translations). The goal of the training is to find a weight vector w such that for each training sentence $\mathbf{S}_i$, the corresponding decoder outputs $\hat{\mathbf{z}} \in V(\mathbf{S}_i)$ which has the maximum BLEU score among all $\mathbf{z} \in V(\mathbf{S}_i)$ based on Eq. 2 . In other words, if $\hat{\mathbf{z}}$ maximizes the scoring function $s_w(\mathbf{z})$, then $\hat{\mathbf{z}}$ also maximizes the BLEU metric. Based on the description, a simple idea is to learn the BLEU score $\mathrm{Bl}(\mathbf{z})$ for each candidate block sequence $\mathbf{z}$. That is, we would like to estimate w such that $s_w(\mathbf{z}) \approx \mathrm{Bl}(\mathbf{z})$. This can be achieved through least squares regression. It is easy to see that if we can find a weight vector w that approximates $\mathrm{Bl}(\mathbf{z})$, then the decoding-rule in Eq. 2 automatically maximizes the BLEU score. However, in our experiments, this simple idea yields poor results. The main reason for its failure is due to the difficulty of estimating $\mathrm{Bl}(\mathbf{z})$ reliably based only on a linear combination of the feature vector as in Eq. 1. Although it is not possible to accurately approximate the BLEU metric using the scoring function $s_w(\mathbf{z})$ defined in Eq. 1, it is still possible to obtain good translation performance. We note that a good decoder does not necessarily employ a scoring function that approximates the BLEU score. Instead, we only need to make sure that the top-ranked block sequence obtained by the decoder scoring function has a high BLEU score. To formulate this idea, we attempt to find a decoding parameter such that for each sentence $\mathbf{S}$ in the training data, sequences in $V(\mathbf{S})$ with the highest BLEU scores should get $s_w(\mathbf{z})$ scores higher than those with low BLEU scores.

Denote by $V_K(\mathbf{S})$ a set of K block sequences in $V(\mathbf{S})$ with the highest BLEU scores. Our decoded result should lie in this set. We call them the "truth". The set of the remaining sequences is $V(\mathbf{S}) - V_K(\mathbf{S})$, which we shall refer to as the "alternatives". We look for a weight vector w that minimize the following training criterion:

$$\hat{w} = \operatorname*{argmin}_{w} \left[\frac{1}{N} \sum_{i=1}^{N} \Phi(w, V_K(\mathbf{S}_i), V(\mathbf{S}_i)) + \lambda w^2 \right] \tag{3}$$

$$\begin{aligned} \Phi(w, V_K, V) &= \frac{1}{K} \sum_{\mathbf{z} \in V_K} \max_{\mathbf{z}' \in V - V_K} \psi(w, \mathbf{z}, \mathbf{z}') \\ \psi(w, \mathbf{z}, \mathbf{z}') &= \phi(s_w(\mathbf{z}), \mathrm{Bl}(\mathbf{z}); s_w(\mathbf{z}'), \mathrm{Bl}(\mathbf{z}')), \end{aligned}$$

where N is the number of training sentence pairs, ϕ is a non-negative real-valued loss function (whose specific choice is not critical for the purposes of this paper),and $\lambda \geq 0$ is a regularization parameter. In our experiments, results are obtained using the following convex loss

$$\phi(s, b; s', b') = (b - b')(1 - (s - s'))_+^2, \tag{4}$$

where $(z)_+ = \max(0, z)$, s is shorthand for $s_w(\mathbf{z})$, b is shorthand for $\mathrm{Bl}(\mathbf{z})$, and s' and b' are defined accordingly. We refer to this formulation as 'Costmargin' (cost-sensitive margin) method: for each input sentence $\mathbf{S}$ the 'Costmargin' $\Phi(w, V_K(\mathbf{S}), V(\mathbf{S}))$ between the 'true' block sequence set $V_K(\mathbf{S})$ and the 'alternative' block sequence set $V(\mathbf{S})$ is maximized. Note that due to the truth and alternative set up, we always have $b > b'$. This loss function gives an upper bound of the error we will suffer if the order of s and s' is wrongly predicted (that is, if we predict $s \leq s'$ instead of $s > s'$). It also has the property that if for the BLEU scores $b \approx b'$ holds, then the loss value is small (proportional to $b - b'$).

A major contribution of this work is a procedure to solve Eq. 3 approximately. The main difficulty is that the search space $V(\mathbf{S})$ covered by the decoder can be extremely large. It cannot be enumerated for practical purposes. Our idea is to replace this large space by a small subspace $\bar{V}_{rel}(\mathbf{S}) \subset V(\mathbf{S})$ which we call *relevant set*. The possibility of this reduction is based on the following theoretical result.

Lemma 1: Let $\psi(w, \mathbf{z}, \mathbf{z}')$ be continuous function of w, and let $\hat{w}$ be a local solution of Eq. 3. Let $\xi_i(w, \mathbf{z}) = \max_{\mathbf{z}' \in V(\mathbf{S}_i) - V_K(\mathbf{S}_i)} \psi(w, \mathbf{z}, \mathbf{z}')$, and define

$$\begin{aligned} \bar{V}_{rel}(\mathbf{S}_i) \quad = \quad \{ \ \mathbf{z}' \in V(\mathbf{S}_i) : \exists \mathbf{z} \in V_K(\mathbf{S}_i) \\ \text{s.t. } \psi(\hat{w}, \mathbf{z}, \mathbf{z}') = \xi_i(\hat{w}, \mathbf{z}) \ \}. \end{aligned}$$

Then $\hat{w}$ is a local minimum of

$$\left[\frac{1}{N} \sum_{i=1}^{N} \Phi(w, V_K(\mathbf{S}_i), \bar{V}_{rel}(\mathbf{S}_i)) + \lambda w^2 \right]. \tag{5}$$

Proof: Let $Q(w) = N^{-1} \sum_{i=1}^{N} \Phi(w, V_K(\mathbf{S}_i), V(\mathbf{S}_i)) + \lambda w^2$. Let $\hat{w}$ be a local solution of (3), then $Q(\hat{w}) \leq Q(w)$ in a neigbhorhood of $\hat{w}$. Due to the continuity, we may choose the neigbhorhood small enough such that $\forall \mathbf{z}' \in \bar{V}_{rel}(\mathbf{S}_i) - V_K(\mathbf{S}_i)$, $\psi(w, \mathbf{z}, \mathbf{z}') < \xi_i(w, \mathbf{z})$. This means that $\Phi(w, V_K(\mathbf{S}_i), \bar{V}_{rel}(\mathbf{S}_i)) = \Phi(w, V_K(\mathbf{S}_i), V(\mathbf{S}_i))$ in this neigbhorhood of $\hat{w}$, implying the lemma. ■

If ϕ is a convex function of w (as in our choice), then we know that the global optimal solution remains the same if the whole decoding space V is replaced by the relevant set $\bar{V}_{rel}$.

In many practical applications, each subspace $\bar{V}_{rel}(\mathbf{S}_i)$ can be significantly smaller than $V(\mathbf{S}_i)$. This is because it only includes those alternatives $\mathbf{z}'$ with score $s_{\hat{w}}(\mathbf{z}')$ close to one of the selected truth. These are the most important alternatives that are easily confused with the truth. Essentially the lemma says that if the decoder works well on these difficult alternatives (relevant points), then it works well on the whole space. The idea is closely related to active learning in standard classification problems, where we selectively pick the most important samples for labeling in order to maximize classification performance. In the active learning setting, as long as we do well on the actively selected samples, we do well on the whole sample space. In our case, as long as we do well on the relevant set, the decoder will perform well.

Strictly speaking, the transformation of the original problem into the new relevant set formulation does not necessarily make the problem easier a priori: the burden is moved from computing a $\max$ over a large collection of alternate hypotheses to an $\arg\max$ over the same collection. Both can be intractable because the loss incorporates a non-decomposable BLEU factor. However, the importance of this transformation is due to the fact that this idea leads to a practical procedure that employs an approximation of relevant set as part of the algorithm. Under appropriate assumptions, using an approximate relevant set ensures the convergence of the algorithm, as shown in the Appendix. Observe that the relevant set depends on the decoder parameter w, and the decoder parameter is optimized on the relevant set. In a practical algorithm, it is necessary to estimate them jointly using an iterative algorithm. The basic idea is to start with a decoding parameter w, and estimate the corresponding relevant set; we then update w based on the relevant set, and iterate this process. The procedure is outlined in Table I, where we use V_{rel} to denote an approximate relevant set that approximates the true relevant set $\bar{V}_{rel}$. We intentionally leave the implementation details of the (*) step and (**) step open. Moreover, in this general algorithm, we do not have to assume that $s_w(\mathbf{z})$ has the form of Eq. 1. A natural question concerning the procedure is its convergence behavior. It can be shown that under mild assumptions, if we pick in (*) an alternative $\tilde{\mathbf{z}}_k \in V(\mathbf{S}) - V_K(\mathbf{S})$ for each $\mathbf{z}_k \in V_K(\mathbf{S})$ ($k = 1, \dots, K$) such that

TABLE I
GENERIC APPROXIMATE RELEVANT SET METHOD

```
for each data point $\mathbf{S}$
    initialize truth set $V_K(\mathbf{S})$ and alternative set $V_{rel}(\mathbf{S})$
for each decoding iteration $l$: $\ell = 1, \cdots, L$
    for each data point $\mathbf{S}$
        select relevant points $\{\tilde{\mathbf{z}}_k\} \in V(\mathbf{S})$ (*)
        update $V_{rel}(\mathbf{S}) \leftarrow V_{rel}(\mathbf{S}) \cup \{\tilde{\mathbf{z}}_k\}$
    update $w$ by solving Eq. 5 approximately (**)
```

$$\psi(w, \mathbf{z}_k, \tilde{\mathbf{z}}_k) = \max_{\mathbf{z}' \in V(\mathbf{S}) - V_K(\mathbf{S})} \psi(w, \mathbf{z}_k, \mathbf{z}'), \tag{6}$$

then the procedure converges to the solution of Eq. 3. Moreover, the rate of convergence depends only on the property of the loss function, and not on the size of $V(\mathbf{S})$. This property is critical as it shows that as long as Eq. 6 can be computed efficiently, then the Approximate Relevant Set algorithm is efficient. Moreover, it gives a bound on the size of an approximate relevant set with a certain accuracy. One result of this kind is presented in the Appendix. It should be pointed out that in practice exact computation of Eq. 6 may not be feasible. For example, in the machine translation application, this requires finding a sentence that optimizes the BLEU score. This optimization can only be performed over a restricted search space containing only a subset of all sentences (e.g., including only local reordering), as detailed in Section IV. In this case, one may either view the solution as an approximation to Eq. 6, or simply regard $V(\mathbf{S})$ as the subset of sentences that can be efficiently searched by the decoder.

The approximate solution of Eq. 5 in (**) can be implemented using stochastic gradient descent (SGD), where we may simply update w as:

$$w \to w - \eta \frac{1}{K} \sum_{k=1}^{K} \nabla_w \psi(w, \mathbf{z}_k, \tilde{\mathbf{z}}_k).$$

The parameter $\eta > 0$ is a fixed constant often referred to as learning rate. For simplicity, here we skip the regularization term (SGD has an implicit regularization effect when the learning rate η is small), which can be easily incorporated. Convergence results can be proved. For simplicity, we skip the analysis. Up to this point, we have not assumed any specific form of the decoder scoring function in our algorithm. We now consider Eq. 1 used in our model:

$$s_w(\mathbf{z}) = w^T \cdot F(\mathbf{z}),$$

TABLE II
COSTMARGIN TRAINING ALGORITHM: $L = 30$ IS THE NUMBER OF TIMES THE TRAINING DATA IS DECODED, $R = 40$ IS THE NUMBER OF ITERATIONS FOR THE SGD TRAINING. THE LEARNING RATE IS SET TO $\eta = 1e - 4$. THE LOSS FUNCTION $l(z', z)$ IS DEFINED IN SECTION IV.

```
for each input sentence $\mathbf{S}_i$, $i = 1, \cdots, N$
    $V_5(\mathbf{S}_i)$ is the oracle truth
    $V_{rel}(\mathbf{S}_i) \leftarrow \{z_0(\mathbf{S}_i)\}$
for each iteration $\ell$: $\ell = 2, \cdots, L$
    Online Training Step:
    initial weight vector $w = w_0$
    for each 'online' iteration $r = 1, \cdots, R$
        for each input sentence $\mathbf{S}_i$ in random order
            find $s = \underset{m \in \{1, \cdots, |V_5(\mathbf{S}_i)|\}}{\text{argmax}} \text{Bl}(z_{im}))$
            find $t = \underset{m \in \{1, \cdots, |V_{rel}(\mathbf{S}_i)|\}}{\text{argmin}} l(z_{is}, z_{im})$
            $w \leftarrow w + \eta \cdot (x_{is} - x_{it}) \cdot (Bl(z_{is}) - Bl(z_{it}))$
    Decoding Step:
    for each input sentence $\mathbf{S}_i$, $i = 1, \cdots, N$
        compute top-scoring block sequence $\tilde{\mathbf{z}}(\mathbf{S}_i)$ and
        update $V_{rel}(\mathbf{S}_i) \leftarrow V_{rel}(\mathbf{S}_i) \cup \{\tilde{\mathbf{z}}(\mathbf{S}_i)\}$
```

where $F(\mathbf{z}) = \sum_{i=1}^{n} f(b_i, o_i, b_{i-1})$. Using this feature representation and the loss function in Eq. 4, we obtain the following Costmargin SGD update rule:

$$w \rightarrow w + \frac{\eta}{K} \sum_{k=1}^{K} \Delta\text{Bl}_k y_k \, (1 - w^T \cdot y_k)_+, \qquad (7)$$
$$\Delta\text{Bl}_k = \text{Bl}(\mathbf{z}_k) - \text{Bl}(\tilde{\mathbf{z}}_k), \; y_k = F(\mathbf{z}_k) - F(\tilde{\mathbf{z}}_k).$$

This method is related to some other formulations proposed in the machine learning literature for solving structured prediction problems such as [10] and [11]. However, the earlier approaches cannot be directly applied to our problem unless modifications are made. In this context, both the general algorithm presented in Table I as well as its analysis in the appendix are novel.

IV. PRACTICAL IMPLEMENTATION OF RELEVANT SET METHOD

In this section, we present implementation details for the online relevant set training algorithm presented in Section III. The Costmargin training algorithm shown in Table II is used to train translation models based on binary features. It is carried out by running $L = 30$ times over the parallel training data each time decoding all of the training sentences and generating a single block sequence which is added to the (approximate) relevant set $V_{rel}(\mathbf{S}_i)$ of each input sentence pair $\mathbf{S}_i$, where $i = 1, \cdots, N = 230\,000$ (training data details are presented in Section VI). A block sequence generated at decoding step ℓ_1 is used in all subsequent training steps $\ell_2 > \ell_1$. The training data after the l-th decoding step is given as $[\, V_5(\mathbf{S}_i) \,,\, V_{rel}(\mathbf{S}_i) \,]_{i=1}^{N}$, where the size $|V_{rel}(\mathbf{S}_i)|$ of the relevant alternative set is l after the l-th decoding step. $V_5(\mathbf{S}_i)$ is the set of the five highest BLEU scoring oracle block sequences that are computed up-front for all training sentence pairs $\mathbf{S}_i$ and are stored separately as described in Section II. Given the training set $[\, V_5(\mathbf{S}_i) \,,\, V_{rel}(\mathbf{S}_i) \,]_{i=1}^{N}$ the highest BLEU scoring block sequence z_{is} for each relevant set $V_{rel}(\mathbf{S}_i)$ is computed. Given the block sequence z_{is} a block sequence z_{it} is computed which minimizes the loss $l(z_{is}, z_{it})$, where $l(z', z) = -(Bl(z') - Bl(z)) \cdot (s(z') - s(z) - 1)$. In Table II, x_{ij} is used as a short-hand notation for the feature vector $F(z_{ij})$ representing the block sequence z_{ij}, where $F(z) = \sum_{i=1}^{n} f(b_i, o_, b_{i-1})$ is defined as previously. Each time the relevant set has been increased by a single element for all input sentences $\mathbf{S}_i$, the weight vector w is trained using the SGD-based online training algorithm described in Section III carrying out $R = 40$ iterations over the relevant sets for each training sentence $\mathbf{S}_i$. The newly generated weight vector w which is used for the subsequent decoding step is trained from scratch, i.e. starting with the zero weight $w_0 = \{0.0\}^M$ and ignoring the weight vectors of any preceding iteration. Since this training step is carried out on a single machine, it dominates the overall computation time. As each iteration adds a single relevant alternative to the set $V_{rel}(\mathbf{S}_i)$, computation time increases with the number of training iterations: the initial model is trained in a few minutes, while training the model after the 30-th iteration may take a few hours for the most complex models. The decoding of the $230\,000$ training sentence pairs is carried out in parallel on 25 64-Bit Opteron machines . Here, monotone decoding is much faster than decoding with block swapping: it takes less than 0.5 hours while the decoding with swapping takes about an hour. Since the training starts with only the parallel training data and a block set, some initial block sequences have to be generated in order to initialize the global model training: for each input sentence a simple bag of blocks translation is generated. For each input interval that is matched by some block b, a single block is added randomly to the bag-of-blocks translation $\mathbf{z}_0(\mathbf{S})$. The order in which the blocks are generated is ignored. For the initial block sequence only block and word identity features are generated, i.e. features of type f_1 and f_2 in Section II. This step does not require the use of a decoder. This way, the initial relevant set for each training sentence contains just a single alternative. After each decoding step, a feature vector representing the top scoring block sequences is written in binary format to disc. This binary representation lists all the indexes of binary features that are active for that block sequence, e.g. several hundred feature indexes are listed for a typical block sequence. In additional experiments a n-best list was generated instead of just using the single top-scoring block sequence. While no improvements in translation performance as measured by the BLEU score were obtained, using the single-best block sequence only is important for the following reasons: 1) decoding time is increased by about a factor of 3 when no n-best lists are generated, 2) the use of n-best lists significantly increases the disc space requirements which is proportional to the size of the relevant set, i.e. assuming the generation of a 100-best list about $3\,000$ feature vectors need to stored per training sentence pair as opposed to at most 30 when generating a single-best block sequence. Even when writing just a single feature vector for each iteration l all the relevant sets cannot be kept in memory and the SGD algorithm requires constant re-reading of feature vectors from disc such that efficient binary reading procedures have been implemented.

Although in order to achieve fast convergence with a theoretical guarantee, we should use Eq. 6 to update the relevant set, in reality, this idea is difficult to implement

TABLE III
PERCEPTRON-STYLE TRAINING ALGORITHM: $L = 30$ IS THE NUMBER OF TIMES THE TRAINING DATA IS DECODED, $R = 40$ IS THE NUMBER OF ITERATIONS FOR THE SGD TRAINING, $N = 1\,043$ IS THE NUMBER OF DEV-SET SENTENCES. THE LEARNING RATE IS SET TO $\eta = 1e - 5$.

```
for each input sentence $\mathbf{S}_i$, $i = 1, \cdots, N$
    $V_{rel}(\mathbf{S}_i) \leftarrow \{z_0(\mathbf{S}_i), z_1(\mathbf{S}_i)\}$
for each iteration $\ell$: $\ell = 3, \cdots, L$
    Online Training Step:
    initial weight vector $w = w_0$
    for each 'online' iteration $r = 1, \cdots, R$
        for each input sentence $\mathbf{S}_i$ in random order
            find $s = \underset{m \in \{1, \cdots, |V_{rel}(\mathbf{S}_i)|\}}{\operatorname{argmax}} \mathrm{Bl}(z_{im})$
            find $t = \underset{m \in \{1, \cdots, |V_{rel}(\mathbf{S}_i)|\}}{\operatorname{argmax}} s_w(z_{im})$
            $w \leftarrow w + \eta \cdot (x_{is} - x_{it})$
    Decoding Step:
    for each input sentence $\mathbf{S}_i$, $i = 1, \cdots, N$
        compute top-scoring block sequence $\tilde{\mathbf{z}}(\mathbf{S}_i)$ and
        update $V_{rel}(\mathbf{S}_i) \leftarrow V_{rel}(\mathbf{S}_i) \cup \{\tilde{\mathbf{z}}(\mathbf{S}_i)\}$
```

because it requires a more costly decoding step. Therefore in Table II, we adopt an approximation, where the relevant set is updated by adding the decoder output at each stage. In this way, we are able to treat the decoding scheme as a black box. One way to approximate Eq. 6 is to generate multiple decoding outputs and pick the most relevant points based on Eq. 6. Since , as mentioned above, the n-best list generation is computationally costly, only a single block sequence is generated for each training sentence pair. Although we are not able to rigorously prove fast convergence rate for this approximation, it works well in practice, as Figure 2 shows. Theoretically this is because points achieving large values in Eq. 6 tend to have higher chances to become the top-ranked decoder output as well.

V. PERCEPTRON-STYLE ALGORITHM AND MER TRAINING

Table III presents a simplification of the relevant set algorithm in Table II. In Section VI-B, this perceptron-style algorithm is compared empirically to the widely used minimum-error-rate (MER) training, carrying out translation experiments using the same decoder and the same decoder parameter setting. The MER training is the Perl implementation provided by the NAACL 2006 SMT workshop [12]. The perceptron algorithm is a *C++* implementation. Both algorithms share the same data presentation for a candidate translation: a low dimensional feature vector $x \in \mathbb{R}^7$ is computed, where the feature components x_j are obtained by summing various probability features over all blocks in given block sequence $z : F(z) = \sum_{i=1}^{n} f(b_i, o, b_{i-1})$, where $f(b_i, o, b_{i-1}) \in \mathbb{R}^7$ is a 'dense' feature representation of the block orientation bigram $f(b_i, o, b_{i-1})$ and x is a shorthand for the feature vector $F(z)$. The actual probability features are defined in Section VI-B. While the MER training explicitly tries to minimize test set level translation error, the perceptron-style algorithm works by iteratively carrying out a simple ranking task on a sentence-by-sentence basis. Apart from using simple perceptron-style weight updates, the algorithm in Table III is also simplified by introducing so-called 'seed' block sequences. No block labeled training data in terms of gold standard phrase-to-phrase translations is needed to initialize the training, i.e. no oracle BLEU computation as described in Section II is needed. Based on the observation that a 'flat' weight vector w already results in surprisingly good translation scores, i.e. in Table VII the perceptron trained weight vector achieves a BLEU score of 44.7 (as shown in line 11) while the flat weight vector w_1 achieves a BLEU score of 43.1, two different translations per input sentence are generated which are used as 'block sequence seeds'. Here, two weight vectors w_1 and w_2 are generated independently of the training data, i.e. a flat weight vector $w_1 = \{0.1\}^7$ and an almost flat vector w_2 which is generated from w_1 by adding a small random number $\delta \in [-0.01, 0.01]$. As shown in Table III, the initial two weight vectors w_1 and w_2 are used to generate two block sequences $z_0(\mathbf{S}_i)$ and $z_1(\mathbf{S}_i)$ per input sentence $\mathbf{S}_i$. These block sequences are used to initialize a simple labeling scheme: for each iteration r and relevant set $V(S_i)$ the online training algorithm computes the s-th decoder output x_{is} with the highest BLEU score and the t-th decoder output x_{it} with the highest translation score. Each time a candidate set $V(S_i)$ is processed, the cost-insensitive perceptron weight vector update $w \leftarrow w + (x_{is} - x_{ir})$ is used to update the weight vector $w \in \mathbb{R}^7$ on this candidate set. The perceptron algorithm iteratively enlarges the candidate set by translating each sentence of the development data $L - 2$ times, where $L = 30$.

The MER training [2] tries to find a parameter vector $w_1^M \in \mathbb{R}^M$ to optimize the development set BLEU score of a log-linear model $Pr(e_1^I | f_1^J)$, where f_1^J is a source sentence of length J and e_1^I is a target sentence of length I and M is the number of probability features. The model is defined as follows:

$$p_{w_1^M}(e_1^I | f_1^J) \quad = \quad \frac{\exp\left[\sum_{m=1}^{M} w_m h_m(e_1^I, f_1^J)\right]}{\sum_{{e'}_1^{I'}} \exp\left[\sum_{m=1}^{M} w_m h_m({e'}_1^{I'}, f_1^J)\right]} \tag{8}$$

where $h_m(e_1^I, f_1^J)$ is a global feature function. In the MER implementation [12] the feature functions $h_m(e_1^I, f_1^J)$ are obtained by summing probability features along the final decoder path. Using a block-based decoder this is equivalent to summing the probability features over the final block sequence, i.e. if a block sequence z represents a translation from the input sentence f_1^J into the target sentence e_1^I then the m-th component of the global vector $F(z)$ is taken as the feature value $h_m(\cdot)$. Here, the sum in the denominator of Eq. 8 is not computed as it is assumed to be constant for a given input sentence f_1^J. A disadvantage of the MER training algorithm is that by updating the weight vector w component-wise carrying out a one-dimensional line search for each dimension of the weight vector, it cannot handle a high-dimensional feature space. Since the BLEU [13] evaluation metric is computed on the test set level, the MER training also optimizes the parameters w_1^M on the test set level. On the other hand, the perceptron algorithm optimizes the weight vector w by computing BLEU on the sentence level only. At least for the feature set in this paper, the results show no degradation in performance due to this simplification.

For the experiments in Section VI-B, the seed block approach is also used in connection with the Costmargin weight vector updates to allow direct comparison of the different training algorithms, i.e. no oracle block sequences are generated for the development data. When handling real-valued 'informative' features the perceptron algorithm performs as well as the Costmargin algorithm which demonstrates the usefulness of the relevant set approach [1]. While the Costmargin, perceptron, and MER training algorithms result in largely comparable performance on a standard Arabic-English translation task, the online training algorithms have the following advantages over the MER training: 1) By not having to generate a computationally expensive n-best list during training decoding speed is increased by a factor 3: an almost identical translation performance is achieved by generating a single top scoring block sequence on each iteration over the training data. Since the online algorithms compute only sentence-level BLEU scores they are conceptually simple and fast. 2) Compared to the MER training, improved translation performance is obtained when including a non-probabilistic feature component into the feature vector x, e.g. the target phrase length. Additionally, in the case that the evaluation criteria used during training and testing differ (cf. Table VIII) the perceptron algorithm outperforms the Costmargin algorithm. 3) The online training algorithm can handle millions of features on top of an already strong baseline system using probabilistic features as shown in Table IX.

VI. Experiments

The discriminatively trained block sequence model for SMT presented in this paper is tested on an Arabic-to-English translation task. The training data consists of the official data available for the NIST 2004 MT evaluation. About $92\,000$ sentences (2.8 million words) come from newswire data, and 3.77 million sentence pairs (115 million words) come from UN proceedings. The language model used during the experiments is trained on IBM proprietary data consisting of 1.15 billion English words. Some punctuation tokenization and some number classing are carried out on the English and the Arabic training data. Translation results in terms of the automatic cased BLEU evaluation metric [13] are presented on the MT03 Arabic-English DARPA evaluation test set consisting of 663 sentences with $16\,264$ Arabic words and 4 reference translations. For some of the experiments using 'specialized' probabilistic features in Section VI-B a development test set is used which is the development data provided by LDC for the 2002 DARPA Arabic-English MT evaluation. This data consists of $1\,043$ sentences with $26\,049$ Arabic words and 4 reference translations as well. A summary of the training and test data statistics is shown in Table IV.

Currently, in order to be able to carry out the discriminative training algorithm the original training data is filtered according to the MT03 test set: all the Arabic substrings up to length 12 that occur in the test set are computed and the parallel training data is sampled to include at least one training sentence pair for each of these phrases. The resulting 'MT03-specific' training data contains about $230\,000$ sentence pairs. Contrary to the pre-filtering of the training data, the block set used in the experiments is not pre-filtered according to any particular test data and consists of about $9\,532\,000$ blocks. Experiments where the block set was also pre-filtered according the MT03 test set resulted in a decreased BLEU score by about 3 %. The block set is derived using a phrase-pair selection algorithm similar to [9], [14]. Blocks that occur only once in the training data might be included as well. Additionally, some heuristic filtering is used to increase phrase translation accuracy. In all the experiments reported, word casing is added as a post-processing step using a statistical model (details are omitted here). Translation performance in terms of BLEU is typically improved by about 2 % ignoring the case information. Translation results in terms of two automatic evaluation metrics for SMT, namely the BLEU evaluation metric [13] and the METEOR evaluation metric [15] are used in this paper. METEOR has been shown to highly correlate with human quality assessment and it can be computed fast which is important for carrying out the decoder-based experiments in section VI-B.

[1]On the contrary, the perceptron algorithm results in significantly lower BLEU scores when used with binary features only.

TABLE IV
TRAINING AND TEST SET STATISTICS FOR THE ARABIC-ENGLISH TRANSLATION EXPERIMENTS PRESENTED IN THIS PAPER.

		Arabic	English
Train	Sentences	229 247	
	Words	5 519 768	6 761 083
DEV MT02	Sentences	1 043	
	Words	26 049	119 176 (4 refs)
TEST MT03	Sentences	663	
	Words	16 264	92 647 (4 refs)

A. Experiments with Binary Features

Table V presents experimental results in terms of uncased BLEU and METEOR for a model that uses binary feature functions. Two reordering restrictions are tested, i.e. monotone decoding ('MON'), and local block reordering where neighbor blocks can be swapped ('SWAP') (see Figure 1 for an example). While the 'SWAP' reordering is quite restrictive, it allows for the decoding step in the training algorithm in Table II to be carried out fast. In order to obtain the results in Table V, the following binary feature types are used which are listed separately according to whether they are defined on the word level or on the phrase level. The following word-level features are used: 'Model 1' type features as defined in Eq. 2, distortion type features, features on target language trigrams, and a special word bigram feature for boundary words of adjacent target phrases. The phrase-based features include block unigram features (as defined in Eq. 2), block orientation features, target and source phrase bigram features, as well as a phrase length features:

$$f_3(b_i, o_i, b_{i-1}) = \begin{cases} 1 & \text{block } b_i \text{ consists of the target phrase 'Lebanese airspace' and the source phrase is 2 words long} \\ 0 & \text{otherwise .} \end{cases}$$

TABLE V
BLEU TRANSLATION RESULTS ON THE TRAINING DATA (230 000 SENTENCES) AND THE MT03 TEST DATA (663 SENTENCES) USING BINARY-VALUED FEATURES.

	Re-ordering	Feature types	training	test	
			training	**test bleu**	**test meteor**
1	'MON'	oracle bleu	54.2	-	-
2		phrase	41.6	28.2	59.1
3		word	42.3	34.0	60.8
4		all	45.8	34.7	61.1
5		all & LM	34.1	36.5	60.4
6	'SWAP'	oracle bleu	59.4	-	-
7		phrase	43.9	30.5	59.7
8		word	44.9	34.4	61.0
9		all	44.4	35.4	61.3
10		all & LM	36.6	38.7	62.9

In addition to the features mentioned above, POS-based features have been tested but did not improve translation performance. For the results with word-based features only, the decoder still generates phrase-to-phrase translations, but all the scoring is done on the word level.

Line 2-4 and line 7-9 in Table V present results for a translation system that is based on binary features only [2]. The fourth column shows translation performance on the training data in terms of averaged sentence-level BLEU score where the length penalty is not taken into account. Column five and six show BLEU and METEOR translation results on the test set level where for the BLEU test score the length penalty is taken into account. The best performing system in line 9 achieves a BLEU score of 35.4 [3]. Line 1 and line 5 of Table V show the averaged sentence-level BLEU score obtained by searching for the highest BLEU scoring block sequence for each training sentence pair as described in Section II. Allowing local block swapping in this search yields a much improved BLEU score of 59.4. The experimental results show that word-based models significantly outperform phrase-based models, the combination of word-based and phrase-based features performs slightly better in terms of BLEU and METEOR translation scores. Swap-based reordering seems to perform slightly better than monotone decoding. For all experiments, the training BLEU score remains significantly lower than the maximum obtainable BLEU score shown in line 1 and line 5. In this respect, there is significant room for improvements in terms of feature functions and alternative set generation. The word-based models perform surprisingly well, i.e. the model in line 8 uses only three feature types: 'Model 1' types features like f_{1001} in Section II, distortion features, and target language m-gram features up to $m = 3$. Training speed varies depending on the feature types used: for the simplest model shown in line 2 of Table V, the training takes about 12 hours, for the models using word-based features shown in line 3 and line 8 training takes less than 2 days. Finally, the training for the most complex model in line 8 takes about 4 days where up to 35 million feature weights are trained.

[2]The results in line 5 and line 10 include a single language model feature as explained below.

[3]With a margin of ± 1.4 %, the differences between the results in line 3-4 and line 8-9 are not statistically significant, but the other BLEU result differences are.

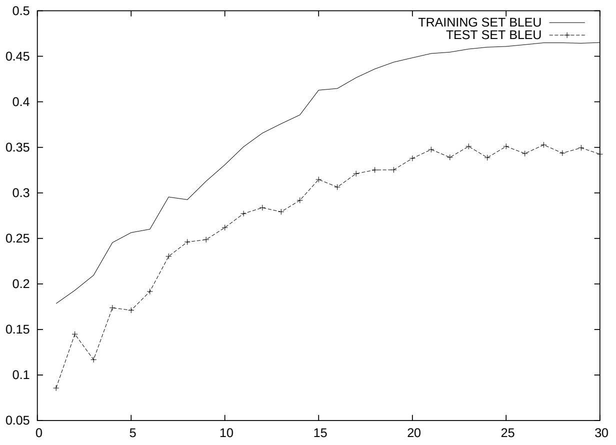

Fig. 2. Training and test learning curves for the Costmargin algorithm in Table II corresponding to line 9 in Table V. The upper graph shows the *averaged* training BLEU scores with a single reference and the lower graph shows the test BLEU scores with four references as a function of the decoding iteration l.

Figure 2 shows the BLEU score for the model corresponding to line 9 in Table V as a function of the number of training iterations: the x axis is the number of training iterations $l = 1, \cdots, L = 30$, the y axis shows the BLEU scores on the training and test data. By adding top scoring alternatives in the training algorithm in Table II, the BLEU score on the training data improves from about 19.0 for the initial model to about 46.0 for the best model after 30 iterations. After each training iteration the test data is decoded as well. Here, the BLEU score improves from 8.0 for the initial model to about 35.5 for the final model (the test data block sequences are not used in the training). The learning curve in Figure 2 is typical for the experiments in Table V: the training BLEU score is much higher than the test set BLEU score despite the fact that the test set uses 4 reference translations. Additionally, line 5 and line 10 in Table V show that a significant improvement in translation quality can be obtained by including a single real-valued feature into the discriminative training, i.e. the trigram language model probability for predicting all the target words in the target phrase of a block (here, feature (c) and (d) from the itemization in Section VI-B are merged into a single feature). The language model feature is included as a component into the global feature vector $F(\mathbf{z}) = \sum_{i=1}^{n} f(b_i, o_i, b_{i-1})$. A constant weight $w_{LM} = 0.1$ is assigned to the language model feature during decoding and training. Figure 3 shows that by including a language model a much faster convergence of the online training is achieved, i.e. the Costmargin algorithm converges after 8 iterations rather than after 25 iterations as shown in Figure 2. The trigram language model does improve translation performance significantly in the 'SWAP case, i.e. from 35.4 to 38.7 in BLEU test score. Interestingly, when using the language model feature the BLEU score on the training data is actually lower while test score is improved: the single language model feature seems prevent the binary model from overfitting on the training data.

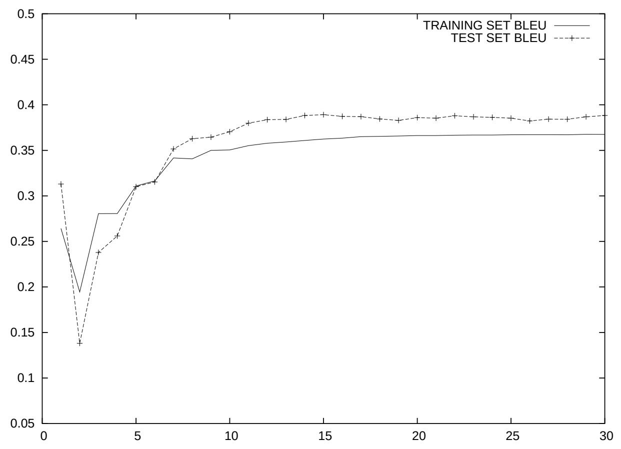

Fig. 3. Training and test learning curves for the Costmargin algorithm in Table II corresponding to line 10 in Table V. When including the LM feature the online training algorithm converges much faster.

TABLE VI
ARABIC-ENGLISH TRANSLATION RESULTS USING THE 6 REAL-VALUED FEATURES (A)-(F) DEFINED IN SECTION VI-B. THE RESULTS ARE OBTAINED WITH A SMALL CONSTANT LEARNING RATE OF $\epsilon = 10^{-5}$.

Line	Training	n-best	training	test	
1	Method	size	**dev bleu**	**test bleu**	**test meteor**
2	flat	-	26.8	40.0	60.8
3	mer	1	32.6	44.6	66.7
4		3	32.7	44.9	66.6
5		10	32.8	44.7	66.7
6		100	32.6	44.9	66.8
7	cost-margin	1	31.2	44.0	67.0
8		3	31.1	44.7	66.6
9		10	31.3	44.0	67.2
10		100	31.4	44.5	67.2
11	percep--tron	1	30.9	44.1	67.0
12		3	30.1	43.7	65.5
13		10	30.9	44.5	66.3
14		100	30.1	42.8	66.5

B. Experiments with Specialized Features: Comparison with MER Training

In this section a detailed empirical comparison between the widely used MER training in [2] and the online training methods presented in this paper is given. As for the minimum error-rate training a Perl implementation of the algorithm is used which has been provided to the participants of the NAACL 2006 Workshop on Statistical Machine Translation [12]. This Perl implementation is modified in such a way that it can be carried out using the same block-based decoder as used for the experiments in Section VI-A. Since the algorithm in Table III is implemented in C++ no direct comparison in terms of CPU time is possible: the algorithms are analyzed in terms of convergence behavior and the size of the n-best list used during the discriminative training. For the experiments in this section a larger block set containing 23 732 807 blocks is used. This block set and the feature components described below are trained on a slightly larger training data consisting of 4.03 million training sentences including some IBM proprietary data. Here, the block set used is not specific to any test data. This has the advantage that development set and test set can be decoded using the same block set. Using test set specific

TABLE VII
ARABIC-ENGLISH TRANSLATION RESULTS USING THE 7 REAL-VALUED FEATURES (A)-(G) DEFINED IN SECTION VI-B, INCLUDING THE 'NON-PROBABILISTIC' (G) FEATURE . HERE, THE MER TRAINING RESULTS IN A LOWER BLEU SCORE.

Line	Training	n-best	training	test	
1	Method	size	**dev bleu**	**test bleu**	**test meteor**
2	flat	-	29.5	43.1	64.7
3	mer	1	30.0	42.5	64.5
4		3	30.6	43.2	65.7
5		10	31.8	44.0	66.7
6		100	31.5	43.7	66.3
7	cost-margin	1	31.9	44.8	66.9
8		3	31.6	44.6	67.6
9		10	32.2	44.9	68.6
10		100	32.1	44.3	68.8
11	percep--tron	1	31.6	44.7	67.0
12		3	31.8	44.6	67.2
13		10	32.1	45.7	67.8
14		100	32.2	44.6	68.7

training data and a test set specific block set for a phrase-based machine translation system typically result in about 2 % BLEU score improvement on the test data: even better translation results can be expected in future experiments using a restricted block set. The following 7 real-valued features are used during the experiments:

- **(a) Direct Model:** the 'direct' translation probability for a block b is defined as $p(b) = N(b = (S,T))/N(T)$, where $N(b)$ is the block unigram count and $N(T)$ is the target phrase unigram count for the target phrase T.
- **(b) Lexical Weighting:** the lexical weight $p(S \mid T)$ of a block $b = (S,T)$ is computed similarly to [9] and depends on the Model 1 translation model probability [16].
- **(c)-(d) Trigram language model:** Two language model features are computed: 1) probability of the first target word in target clump T_i for block b_i given the final two words of the predecessor target clump T_{i-1}, and 2) probability of predicting the rest of the target words in target clump T_i.
- **(e)-(f) Distortion Weighting:** A block reordering model with word-based distortion probabilities is used. The distortion model is computed from word-aligned training data [17]. Similar to the block orientation model, so-called inbound and outbound scores are assigned to a block pair (b_{i-1}, b_i) based on their source positions relative to each other.
- **(g) Target Phrase Length:** The negative number of words in the target phrase T of a block b. This feature is used to control the total number of words in the final translation.

C. Effect of Feature Choice

For comparison purposes, experiments are carried out with different sub-sets of the 7 features defined above. The results in Table VI are obtained using the 6 features (a)-(f), while for the results in Table VII the non-probabilistic (g) feature is used additionally. Results for three algorithms are presented: the Costmargin algorithm presented in Table II, the perceptron

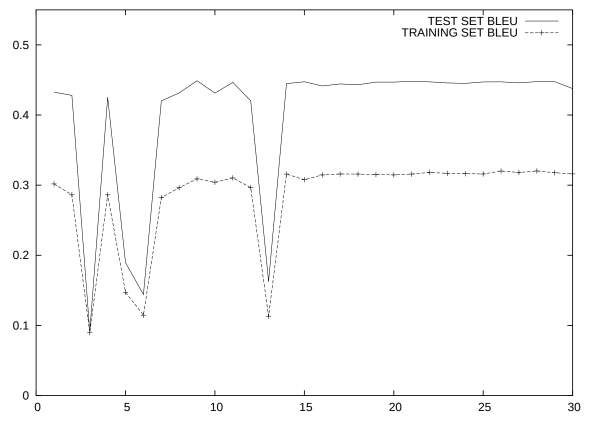

Fig. 4. Training and test learning curves for the Costmargin algorithm in Table II corresponding to line 7 in Table VII. After the two 'seed' block translations have been generated, performance drops until about the 8-th iteration when appropriate alternatives have been generated to train a good weight vector.

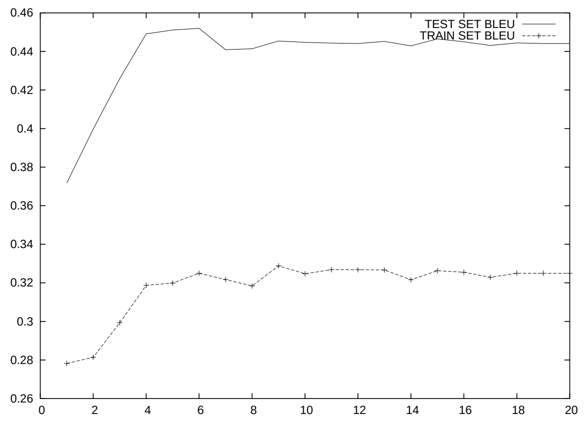

Fig. 6. Training and test learning curves for the MER algorithm corresponding to line 3 in Table VI. Unlike the learning curves presented in Figure 4 or Figure 5, good performance is already achieved after the first few decoding steps.

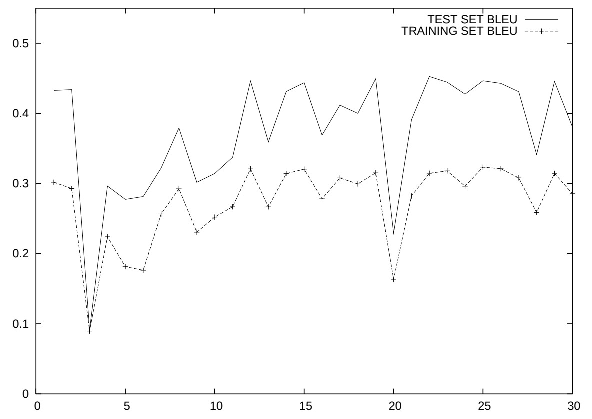

Fig. 5. Training and test learning curves for the perceptron algorithm in Table III corresponding to line 11 in Table VII. The cost-insensitive perceptron update results in a less smooth learning curve, but translation performance is nearly identical.

algorithm shown in Table III, and the MER training. When using the Costmargin algorithm, the block sequence 'seed' simplification from Section V is employed, i.e. the training is initialized with two 'flat' weight vectors as for the perceptron algorithm. All three algorithms train the same low dimensional weight vector $w \in \mathbb{R}^M$, where $M \in \{6, 7\}$. In both tables, the second column reports the training algorithm used. Line 2 shows the surprisingly good BLEU score obtained when decoding with a 'flat' weight vector $w_0 = \{0.1\}^M$. The third column shows the size of the n-best list, the fourth column reports the averaged sentence-level training BLEU score [4], the fifth and sixth column show the MT03 test set BLEU and METEOR scores [5]. The three algorithms are analyzed in terms of translation performance and size of the n-best list. N-best lists are generated using the algorithm in [18]. [19] gives details on the block-based decoder and the translation graph generation.

Figure 4, and Figure 5, and Figure 6 show learning curves on training and test data for the three different algorithms. Here, the x axis is the number of training iterations $l \in \{1, \cdots, L = 30\}$ ($L = 20$ for the MER training), and the y axis is the BLEU performance on the training and test data. In Figure 4, the averaged sentence-level BLEU score on the training data is improved from 29.5 for the initial model to 31.9 after 30 iterations. Simultaneously, on the MT03 test set, the BLEU score improves from 43.1 to 44.8. The learning curve in Figure 4 is typical for the experiments in Table VI and in Table VII: it is less smooth than the one shown in Figure 2 where only binary feature are used. This is particular true for the cost-insensitive perceptron weight vector updates as shown in Figure 5. To obtain the scores shown in Table VI and Table VII the online training algorithm is carried out to obtain a weight vector w_l for each iteration l. The weight vector w_l which corresponds to the highest development set BLEU score is used to decode the MT03 test set as well. Although the learning curves corresponding to the perceptron training are not as smooth as the ones obtained from the Costmargin training, after carrying out $L = 30$ iterations, translation performance does not seem to be effected significantly. Figure 6 shows the learning curve corresponding to the MER training. Even so the MER training conceptually maximizes the corpus-level BLEU score the averaged sentence-level BLEU score is reported for direct comparison purposes. The MER learning curve is much smoother, but the MER training algorithm is more complex than the online training algorithms which need only a few seconds of CPU time to run on the small development set. Contrary to [2] where the use of a 200-best list is reported, the results in Table VI and in Table VII show that a good translation performance can be obtained using a 1-best list even in the case of the MER training. In the case of the Costmargin and the perceptron algorithm an almost identical performance is obtained using a 1-best list. When using 6 real-valued features, Table VI shows that the MER training obtains the best BLEU score on training and test data. Although the MER training achieves a significantly

[4]While the averaged sentence-level BLEU scores are around 30 % , the corpus-level BLEU scores on the MT02 development set are in the same range as the MT03 corpus-level BLEU scores, i.e. around 44.0 %.

[5]The margin for significant BLEU differences is about ± 1.5 % on the MT03 test set.

higher BLEU score on the training data (a BLEU score of 32.8 (line 5) vs. a BLEU score of 31.4 (line 10) for the Costmargin training), the performance difference on the test data is much smaller (44.7 vs. 44.5), i.e. while the MER-trained weight vector fits the training data better no advantage is obtained on the test data. Moreover, when including a non-probabilistic feature the MER training achieves a significantly lower BLEU score as shown in Table VII. Here, performance can be improved significantly by increasing the size of the n-best list. On the other hand, the Costmargin and the perceptron algorithm achieve a close to optimal BLEU score by producing a 1-best list only. Here, the perceptron and the Costmargin training algorithms have a nearly identical run time which is not effected by the different weight vector updates. A typical run over the MT02 development data takes about 500 seconds on a single Opteron machine (using a trigram language model). Generating an n-best list slows down decoding by about a factor of 3, where the size n of the n-best list does not effect the decoding time significantly. Online training time is only about 0.43 seconds per iteration, i.e. less than 1 % percent of the total training time is needed for the online training step in Table II and in Table III. The online training time is close to being linear in the size of the n-best list as can be expected from the description of the algorithm in Table III: the complexity of the argmax operations in the inner-most loop is linear in the size $|V(\mathbf{S})|$ of the relevant set. Consequently when using a 10-best list training time is increased to 3.5 seconds per online iteration. Using a 5-gram language model in the experiments corresponding to line 7 and line 11 in Table VII improves the BLEU test score by about 2.5 % to 47.3 and 46.9 respectively. Since using a 5-gram language model increases decoding time by more than a factor 10 only a 3-gram language model is used for the experiments in this paper.

D. *Effect of Evaluation Metric*

Table VIII shows the effect of using a different evaluation metric during training. Here, the multi-reference word-error rate (mWER) [20] replaces the averaged sentence-level BLEU score during training. The mWER is based on a simple word-matching algorithm and can be computed efficiently. In order to apply the Costmargin training algorithm, a reverse mWER $\epsilon^{'}$ is defined: $\epsilon^{'} = 1 - \epsilon$, where ϵ is the regular mWER which is computed on a sentence-by-sentence basis. The reverse mWER is used as a replacement for the BLEU score in the Costmargin algorithm. Again, experiments are carried out for different sizes n of the n-best list. The results in Table VIII show that using the reverse mWER as a training criterion decreases BLEU and METEOR test scores when using the Costmargin training algorithm: this is due the evaluation metric mis-match on training and test data. On the other hand, for the cost-insensitive perceptron algorithm the performance degradation is minor compared to the results in Table VII.

E. *Combination of Real-valued and Binary-valued Features*

Table IX demonstrates the improvement obtained from including binary-valued features on top of a baseline system using the 7 real-valued features (a)-(g) described in Section VI. The baseline system is the one corresponding to line 7 in Table VII, i.e. the weight vector is trained using the Costmargin algorithm with a 1-best list. The training is carried out in two stages: either the weight for the real-valued features is trained first and kept fixed while the weights for the binary features are trained or the binary feature weights are trained first followed by the training of the weights for the real-valued features.

The binary feature weights are trained on the parallel training data consisting of the 230 thousand sentence pairs used for the experiments in Section VI-A. The weight vector for the real-valued feature components is trained on the MT02 development set. Table VII shows that a small improvement in BLEU and METEOR scores is obtained on top of an already good baseline system, i.e. the MT03 BLEU score is improved from 44.8 to 45.6 and the METEOR score is improved from 66.9 to 67.8 (line 3 of Table IX). These results are obtained by training the binary feature weights first. Training the real-valued feature weights first (i.e. re-using the weights from line 1) results in a worse BLEU score as shown in line 2. Training the weights for both binary and real-valued features simultaneously is future work as the current training is unstable in this case.

TABLE VIII
ARABIC-ENGLISH TRANSLATION RESULTS USING THE MULTIPLE WORD-ERROR-RATE (MWER) AS TRAINING CRITERION. HERE, THE PERCEPTRON TRAINING RESULTS IN SUBSTANTIALLY BETTER BLEU TEST SCORES.

Method	n-best	training	test	
	size	**dev mwer**	**test bleu**	**test meteor**
cost-margin	1	38.2	42.9	66.3
	3	37.9	41.5	64.4
	10	38.2	41.9	65.4
	100	38.2	41.7	65.2
percep-tron	1	38.0	45.3	66.4
	3	36.4	44.3	65.5
	10	37.8	44.5	66.6
	100	36.7	44.3	65.6

TABLE IX
ARABIC-ENGLISH TRANSLATION RESULTS DEMONSTRATING THE USE OF BINARY FEATURES TO IMPROVE A BASELINE SYSTEM BASED ON 7 REAL-VALUED FEATURES.

Line	Set of Features	training	test	
		dev bleu	**test bleu**	**test meteor**
1	7 features	31.9	44.8	66.9
2	7 features & binary	31.6	45.1	67.8
3	binary & 7 features	31.7	45.6	67.8

VII. COMPARISON AND FUTURE WORK

The work in this paper substantially differs from previous work in phrase-based SMT [9], [21], [22] which is based on components which are estimated using generative models similar to the noisy channel approach in [16]. While error-driven training techniques like the MER training algorithm [2] are commonly used, this paper presents a sequential block segmentation approach to SMT that is similar to part-of-speech tagging or shallow parsing. The novel approach treats the decoding process as a black box and is capable of optimizing tens of millions of parameters automatically, which makes it applicable to other problems as well. Although at this stage the system performance based on binary feature functions is not yet better than previous approaches, improvements can be expected in the future, e.g. training weights for phrase-based features discriminatively might have advantages over training phrase-based weights generatively as in [23]. The choice of our formulation is convex. However, the loss function in Eq. 4 may not be optimal, and using different choices may lead to future improvements. Another important direction for performance improvement is to design methods that better approximate Eq. 6. While the global training approach presented in this paper is simple, after 15 iterations or so, as can be seen from the learning curves in Figure 2 and Figure 3, the alternatives that are being added to the relevant set differ very little from each other, slowing down the training considerably such that the set of possible block translations $V(\mathbf{S})$ might not be fully explored. As is shown extensively in Section VI-B the current approach is able to handle models based on real-valued features as well. The perceptron algorithm presented in Table III is conceptually simple, fast and achieves performance comparable to the MER training algorithm.

Two perceptron-like algorithms that handle global features in the context of re-ranking are presented in [24] where results comparable to the MER training are achieved. The results in that paper are based on re-ranking n-best lists and the run time complexity of these algorithms is quadratic in the size n of the n-best lists, where $n = 1000$. On the other hand, the complexity of the online algorithms in this paper is only linear in the size of the alternative set $|V(S)|$. Another perceptron-style discriminative training algorithm for SMT is presented in [25]. Unlike the current work, the perceptron algorithm in [25] is restricted to handle only source sentences of length 5-15 words, and it still uses so-called blanket probability features in all its experiments. Additionally, [25] raises the issue whether a reference translation is reachable for a given block set and decoder and modifies its update strategy accordingly. The problem of reachability within the Costmargin approach is handled by the fact that the gold standard block sequence is generated by the same decoder that is used during training, i.e. the updates are carried out towards a block sequences which is guaranteed to be reachable. On the other hand, when training weights for real-valued features the perceptron algorithm performs as well as the MER training without taking reachability concerns into account. The use of 'seed' block sequences in Section V is related to the use of the 'seed' rules in [26]. The direct translation model presented in [27] can also be compared to the current paper. A MaxEnt model is trained which is capable of handling millions of features. This MaxEnt model still relies on a set of specialized features and binary features are manually crafted to fit the Arabic English direct translation model. In contrast, the discriminative training algorithm presented in this paper treats the decoder as a block box and allows features to be more freely defined.

APPENDIX

We present an instance of the generic approximate relevant set method in Table I, for which we can prove a convergence bound.

TABLE X
APPROXIMATE RELEVANT SET METHOD WITH EXACT MINIMIZATION

```
initialize weight vector $w_0 \leftarrow 0$
**for each** data point $\mathbf{S}_i$
    initialize truth $V_K(\mathbf{S}_i)$ and alternative $V_0^{(r)}(\mathbf{S}_i) \leftarrow \{\}$
**for each decoding iteration** $\ell$: $\ell = 1, \cdots, L$
    let $V_\ell^{(r)}(\mathbf{S}_i) \leftarrow V_{\ell-1}^{(r)}(\mathbf{S}_i)$
    **for each** data point $\mathbf{S}_i$ and $\mathbf{z}_k \in V_K(\mathbf{S}_i)$
        select an alternative $\tilde{\mathbf{z}}_k^\ell(\mathbf{S}_i) \in V(\mathbf{S}_i) - V_K(\mathbf{S}_i)$ such that
        $\psi(w_{\ell-1}, \mathbf{z}_k, \tilde{\mathbf{z}}_k^\ell(\mathbf{S}_i)) = \max_{\mathbf{z}' \in V(\mathbf{S}_i) - V_K(\mathbf{S}_i)} \psi(w_{\ell-1}, \mathbf{z}_k, \mathbf{z}')$
        update $V_\ell^{(r)}(\mathbf{S}_i) \leftarrow V_\ell^{(r)}(\mathbf{S}_i) \cup \{\tilde{\mathbf{z}}_k^\ell(\mathbf{S}_i)\}$
    let $Q_\ell(w) = N^{-1} \sum_{i=1}^N \Phi(w, V_K(\mathbf{S}_i), V_\ell^{(r)}(\mathbf{S}_i)) + \lambda w^2$ and
    update $w$: $w_\ell \leftarrow \arg\min_w Q_\ell(w)$
```

Lemma 2: Consider Table X, and assume that $\Phi(w, V, V')$ is a convex function of w. Let

$$M_\ell = \sup_{i, \mathbf{z}_k \in V_K(\mathbf{S}_i)} \|\nabla_w \psi(w_{\ell-1}, \mathbf{z}_k, \tilde{\mathbf{z}}_k^\ell(\mathbf{S}_i)) + \lambda w\|_2.$$

Define $Q(w) = N^{-1} \sum_{i=1}^N \Phi(w, V_K(\mathbf{S}_i), V(\mathbf{S}_i)) + \lambda w^2$, and $\delta_\ell = \inf_w Q(w) - Q_\ell(w_\ell)$, then

$$0 \le \delta_\ell \le \max(0.5\delta_{\ell-1}, \delta_{\ell-1} - 0.25\lambda M_\ell^{-2} \delta_{\ell-1}^2).$$

Moreover, let $\hat{w} = \arg\min_w Q(w)$, then $\|w_\ell - \hat{w}\|_2 \le \sqrt{\delta_\ell/\lambda}$.

Proof: Since for each w, $Q(w) \ge Q_\ell(w) \ge Q_{\ell-1}(w)$, we have the following inequalities:

$$Q_\ell(w_{\ell-1}) = Q(w_{\ell-1}) \ge \inf_w Q(w) \ge Q_{\ell-1}(w_{\ell-1}).$$

Therefore if we let $d_\ell = Q_\ell(w_{\ell-1}) - Q_{\ell-1}(w_{\ell-1})$, then $\delta_{\ell-1} \le d_\ell$. Let $b_\ell = Q_\ell(w_\ell) - Q_{\ell-1}(w_{\ell-1})$, then $b_\ell = \delta_{\ell-1} - \delta_\ell$. Since $\nabla_w Q_{\ell-1}(w_{\ell-1}) = 0$, we can use convexity and obtain

$$\begin{aligned} b_\ell =& Q_\ell(w_\ell) - Q_{\ell-1}(w_{\ell-1}) \\ \ge& Q_{\ell-1}(w_\ell) - Q_{\ell-1}(w_{\ell-1}) \\ \ge& Q_{\ell-1}(w_\ell) - Q_{\ell-1}(w_{\ell-1}) - \nabla_w Q_{\ell-1}(w_{\ell-1})^T (w_\ell - w_{\ell-1}) \\ \ge& \lambda \|w_{\ell-1} - w_\ell\|_2^2. \end{aligned}$$

Similarly, by convexity,

$$Q_\ell(w_{\ell-1}) - Q_\ell(w_\ell) \le \nabla_w Q_\ell(w_{\ell-1})^T (w_{\ell-1} - w_\ell).$$

Therefore

$$\begin{aligned} d_\ell - b_\ell =& Q_\ell(w_{\ell-1}) - Q_\ell(w_\ell) \\ \le& |\nabla_w Q_\ell(w_{\ell-1})^T (w_{\ell-1} - w_\ell)| \\ \le& M_\ell \|w_{\ell-1} - w_\ell\|_2 \\ \le& M_\ell \sqrt{b_\ell/\lambda}. \end{aligned}$$

Now using $\delta_{\ell-1} \le d_\ell$ and $b_\ell = \delta_{\ell-1} - \delta_\ell$, we have $\delta_\ell \le d_\ell - b_\ell \le M_\ell \sqrt{(\delta_{\ell-1} - \delta_\ell)/\lambda}$. That is,

$$\delta_\ell \le \delta_{\ell-1} - \lambda M_\ell^{-2} \delta_\ell^2.$$

We thus either have $\delta_\ell \le 0.5\delta_{\ell-1}$ or $\delta_\ell \le \delta_{\ell-1} - 0.25\lambda M_\ell^{-2} \delta_{\ell-1}^2$. This proves the first claim. For the second claim, we note that the first order condition $\nabla_w Q_\ell(w_\ell) = 0$ holds. Therefore by convexity, we have

$$\begin{aligned} & Q_\ell(\hat{w}) - Q_\ell(w_\ell) \\ = \;& Q_\ell(\hat{w}) - Q_\ell(w_\ell) - \nabla_w Q_\ell(w_\ell)^T(\hat{w} - w_\ell) \\ \geq \;& \lambda \|w_\ell - \hat{w}\|_2^2. \end{aligned}$$

This implies that $\delta_\ell = Q(\hat{w}) - Q_\ell(w_\ell) \geq \lambda\|w_\ell - \hat{w}\|_2^2$. ■

Lemma 3: Assume that for $\ell \geq 1$,

$$\delta_\ell \leq \max(0.5\delta_{\ell-1}, \delta_{\ell-1} - 0.25\lambda M_\ell^{-2}\delta_{\ell-1}^2).$$

Let $A = 2\sup_{\ell \leq L} M_\ell^2/\lambda$ and $\ell_* = \max(0, \log_2(\delta_0/A))$. Then when $\ell \geq \ell_*$,

$$\delta_\ell \leq 2A/(\ell - \ell_* + 2).$$

Proof: If $\delta_\ell \geq A$, then $\delta_\ell \leq 0.5\delta_{\ell-1} \cdots \leq 0.5^\ell\delta_0$. This implies that $\delta_{\ell_*} \leq A$. We now prove the lemma by induction. When $\ell = \ell_*$, the claim holds. When $\ell > \ell_*$, we assume that the claim holds for $\delta_{\ell-1}$, then

$$\begin{aligned} \delta_\ell \leq & \delta_{\ell-1} - \delta_{\ell-1}^2/(2A) \\ \leq & 2A/(\ell - \ell_* + 1) - 2A/(\ell - \ell_* + 1)^2 \\ = & 2A[(\ell - \ell_*)(\ell - \ell_* + 2)/(\ell - \ell_* + 1)^2]/(\ell - \ell_* + 2) \\ \leq & 2A/(\ell - \ell_* + 2). \end{aligned}$$

This proves the lemma for δ_ℓ. ■

Theorem 1: Under the conditions of Lemma 2, and let $A = 2\sup_{\ell \leq L} M_\ell^2/\lambda$ and $\ell_* = \max(0, \log_2(Q_1(0)/A))$. Then when $\ell \geq \ell_*$, $\delta_\ell \leq 2A/(\ell - \ell_* + 2)$ and

$$\|w_\ell - \hat{w}\|_2 \leq \sqrt{2A/\lambda(\ell - \ell_* + 2)}.$$

Proof: We note that $\delta_0 = \inf_w Q(w) - Q_0(0) \leq Q_1(0) - Q_0(0) = Q_1(0)$. The claims are now direct consequences of Lemma 2 and Lemma 3. ■

REFERENCES

[1] C. Tillmann and T. Zhang, "A Discriminative Global Training Algorithm for Statistical MT," in *Proceedings of ACL-COLING'06*, Sydney, Australia, July 2006, pp. 721–728.

[2] F. J. Och, "Minimum Error Rate Training in Statistical Machine Translation," in *Proceedings of ACL'03*, Sapporo, Japan, 2003, pp. 160–167.

[3] C. Tillmann and T. Zhang, "A Localized Prediction Model for Statistical Machine Translation," in *Proceedings of ACL'05*. Ann Arbor, MI: Association for Computational Linguistics, June 2005, pp. 557–564.

[4] S. Kumar and W. Byrne, "Local Phrase Reordering Models for Statistical Machine Translation," in *Proceedings of HLT/EMNLP'05*, Vancouver, British Columbia, Canada, October 2005, pp. 161–168.

[5] R. McDonald, K. Crammer, and F. Pereira, "Online Large-Margin Training of Dependency Parsers," in *Proceedings of ACL'05,* Ann Arbor, MI, June 2005, pp. 91–98.

[6] M. Collins, "Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms," in *Proceedings of EMNLP'02,* Philadelphia,PA, July 2002, pp. 1–8.

[7] M. Nagata, K. Saito, K. Yamamoto, and K. Ohashi, "A Clustered Global Phrase Reordering Model for Statistical Machine Translation," in *Proceedings of ACL-COLING'06*, Sydney, Australia, July 2006, pp. 713–720.

[8] R. Zens and H. Ney, "Improvements in Phrase-Based Statistical Machine Translation," in *HLT-NAACL 2004: Main Proceedings,* Boston, MA, USA, May 2 - May 7 2004, pp. 257–264.

[9] P. Koehn, F. J. Och, and D. Marcu, "Statistical Phrase-Based Translation," in *HLT-NAACL 2003: Main Proceedings,* Edmonton, Alberta, Canada, May 27 - June 1 2003, pp. 127–133.

[10] B. Taskar, C. Guestrin, and D. Koller, "Max-Margin Markov Networks," in *Advances in Neural Information Processing Systems 16*, S. Thrun, L. Saul, and B. Schölkopf, Eds. Cambridge, MA: MIT Press, 2004.

[11] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large Margin Methods for Structured and Interdependent Output Variables," *JMLR*, vol. 6, pp. 1453–1484, 2005.

[12] Philipp Koehn and Christof Monz, Workshop Overview Paper of the NAACL 2006 Workshop on SMT, New York, NY, June 2006.

[13] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: A Method for Automatic Evaluation of Machine Translation," in *Proceedings of ACL'02*, Philadelphia, PA, July 2002, pp. 311–318.

[14] C. Tillmann, "A Projection Extension Algorithm for Statistical Machine Translation," in *Proceedings of EMNLP'03*, Sapporo, Japan, July 2003, pp. 1–8.

[15] S. Banerjee and A. Lavie, "METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments," in *Proc. of the Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization at ACL'05,* Ann Arbor, MI, June 2005.

[16] P. F. Brown, V. J. Della Pietra, S. A. Della Pietra, and R. L. Mercer, "The Mathematics of Statistical Machine Translation: Parameter Estimation," *Computational Linguistics*, vol. 19, no. 2, pp. 263–311, 1993.

[17] Y. Al-Onaizan and K. Papineni, "Distortion Models for Statistical Machine Translation," in *Proceedings of ACL-COLING'06,* Sydney, Australia, July 2006, pp. 529–536.

[18] F.-J. Och, N. Ueffing, and H. Ney, "An Efficient A* Search Algorithm for Statistical Machine Translation," in *Proc. of Data-Driven MT Workshop at ACL'01*, Toulouse, France, July 2001, pp. 55–62.

[19] C. Tillmann, "Efficient Dynamic Programming Search Algorithms for Phrase-based SMT," in *Proceedings of the Workshop CHPSLP at HLT'06*, New York City, NY, June 2006, pp. 9–16.

[20] S. Niessen, F.-J. Och, G. Leusch, and H. Ney, "An Evaluation Tool for Machine Translation: Fast Evaluation for MT Research," in *Proc. of the 2nd Int. Conf. on Language Resources and Evaluation,* Athens, Greece, May 2000, pp. 39–45.

[21] F.-J. Och, C. Tillmann, and H. Ney, "Improved Alignment Models for Statistical Machine Translation," in *Proc. of 'EMNLP/VLC'99*, College Park, MD, June 1999, pp. 20–28.

[22] D. Marcu and D. Wong, "A Phrase-Based, Joint Probability Model for Statistical Machine Translation," in *Proceedings of EMNLP'02,* Philadelphia, July 2002, pp. 133–139.

[23] J. DeNero, D. Gillick, J. Zhang, and D. Klein, "Why Generative Phrase Models Underperform Surface Heuristics?" in *Proceedings on the Workshop on SMT*, New York City, June 2006, pp. 31–38.

[24] L. Shen, A. Sarkar, and F.-J. Och, "Discriminative Re-ranking of Machine Translation," in *Proceedings of the Joint HLT and NAACL Conference (HLT'04),* Boston, MA, May 2004, pp. 177–184.

[25] P. Liang, A. Bouchard-Côté, D. Klein, and B. Taskar, "An End-to-End Discriminative Approach to Machine Translation," in *Proceedings of ACL-COLING'06*, Sydney, Australia, July 2006, pp. 761–768.

[26] M. Collins and Y. Singer, "Unsupervised Models for Named Entity Classification," in *Proceedings of EMNLP'99*, College Park,MD, June 1999, pp. 100–110.

[27] A. Ittycheriah and S. Roukos, "Direct Translation Model 2," in *Main Proceedings of HLT'02*, Rochester, New York, April 2007, pp. 57–64.