A Block Bigram Prediction Model for Statistical Machine Translation

CHRISTOPH TILLMANN IBM T.J. Watson Research Center TONG ZHANG Yahoo Research

In this paper, we present a novel training method for a localized phrase-based prediction model for statistical machine translation (SMT). The model predicts block neighbors to carry out a phrasebased translation that explicitly handles local phrase re-ordering. We use a maximum likelihood criterion to train a log-linear block bigram model which uses real-valued features (e.g. a language model score) as well as binary features based on the block identities themselves (e.g. block bigram features). The model training relies on an efficient enumeration of local block neighbors in parallel training data. A novel stochastic gradient descent (SGD) training algorithm is presented that can easily handle millions of features. Moreover, when viewing SMT as a block generation process, it becomes quite similar to sequential natural language annotation problems such as part-of-speech tagging, phrase chunking, or shallow parsing. The novel approach is successfully tested on a standard Arabic-English translation task using two different phrase re-ordering models: a block orientation model and a phrase-distortion model.

Categories and Subject Descriptors: I.2.7 [Artificial Intelligence]: Natural Language Processing—statistical machine translation; G.3 [Probability and Statistics]: Statistical computing stochastic gradient descent

General Terms: Algorithms, Experimentation, Languages, Theory

Additional Key Words and Phrases: Statistical machine translation, machine learning, maximum entropy, stochastic gradient descent

1. INTRODUCTION

Statistical machine translation was pioneered by the IBM Candide machine translation project. Their noisy channel approach is presented in [Brown et al. 1993]. In this framework, the units of translation are single words and word-based translation and distortion probabilities are trained automatically from parallel data using the EM training algorithm. In recent years, so-called phrase-based machine translation

Author's address: Christoph Tillmann, IBM T.J. Watson Research Center, Yorktown Heights, NY 10598; email: ctill@us.ibm.com. Tong Zhang, Yahoo Research, New York City; email: tzhang@yahoo-inc.com.

A preliminary version of this work was published as [Tillmann and Zhang 2005].

Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.

© 2007 ACM 0000-0000/2007/0000-0001 \$5.00

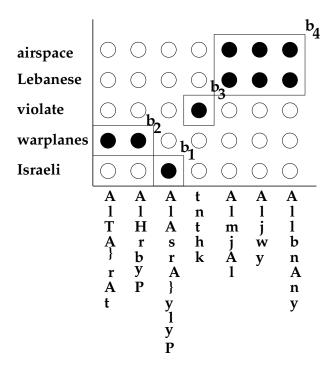


Fig. 1. An Arabic-English block translation example, where the Arabic words are romanized. The following orientation sequence is generated: $o_1 = N$, $o_2 = L$, $o_3 = N$, $o_4 = R$.

approaches have become popular because they generally showed better translation results on standard evaluation corpora. Several approaches to phrase-based SMT are extensions of the noisy channel approach: the alignment-template approach [Och et al. 1999; Och and Nev 2004] is formulated as a phrase-based extension of [Brown et al. 1993]. Other examples of research on phrase-based MT based on the noisy channel model are the example-based model in [Marcu 2001], the joint model presented in [Marcu and Wong 2002], the HMM-based phrase translation models in [Deng and Byrne 2005; Zens and Nev 2004], and the chunk-based model presented in [Watanabe et al. 2003]. The work on weighted finite-state transducers in [Kumar and Byrne 2003; 2005] and the shallow syntactic parsing approach presented in [Schafer and Yarowsky 2003] are also based on the noisy channel model. On the other hand, there is a series of papers that emphasize the need to extract or compute efficiently high quality phrase-based translation lexica [Callison-Burch et al.; Koehn et al. 2003; Tillmann 2003; Venugopal et al. 2003; Zhao et al. 2005]. There have also been successful attempts to combine phrase-based machine translation and syntactic parsing [Chiang 2005; Chiang et al. 2005].

This paper presents a quite different view of phrase-based SMT: the translation is carried out by generating so-called 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 4 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 monotone block sequence is generated except for the possibility to handle some local phrase re-ordering. For illustrating the novel training approach, we use an orientation model similar to the lexicalized block re-ordering model in [Tillmann 2004; Och et al. 2004; Kumar and Byrne 2005]: a block b with orientation o is generated relative to its predecessor block b'. During decoding, we compute the probability $P(b_1^n, o_1^n)$ of a block sequence b_1^n with orientation o_1^n as a product of block bigram probabilities:

$$P(b_1^n, o_1^n) \approx \prod_{i=1}^n p(b_i, o_i | b_{i-1}, o_{i-1}),$$
 (1)

where b_i is a 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 (the orientation o_{i-1} of the predecessor block is ignored.), and n is number of blocks in the translation. Here, the block sequence with orientation (b_1^n, o_1^n) is generated under the restriction that the concatenated source phrases of the blocks b_i yield the input sentence. In modeling a block sequence, we emphasize adjacent block neighbors that have **Right** or **Left** orientation. Blocks with neutral orientation are supposed to be less strongly 'linked' to their predecessor block and are handled separately. During decoding, most blocks have right orientation (o = R), since the block translations are mostly monotone. In this paper, swapping of local block neighbors is the only possible phrase re-ordering considered by the model. Neutral orientation blocks occur only in very restricted cases during decoding while during training only left and right orientation for adjacent blocks is handled in the enumeration scheme presented in Section 5.1. Because of the very restricted phrase re-ordering used for the experiments in that section, more sophisticated orientation label schemata are not investigated in this paper (See Section 6 for a more detailed discussion on additional block orientation schemata.).

In Section 5.3, the paper presents a modified block sequence model which does not use block orientation features, i.e. a block sequence is not labeled according to an orientation scheme as in Fig 1. The model allows for less restrictive block re-ordering which results in a slower decoding step. Additionally, the alternative generation step to successfully train the model is more complicated. BLEU performance is improved due to a richer feature set.

The focus of this paper is to investigate issues in discriminative training of decoder parameters. Instead of directly minimizing error as in earlier work [Och 2003], we decompose the decoding process into a sequence of local decision steps based on Eq. 1, and then train each local decision rule using convex optimization techniques ¹. The advantage of this approach is that it can easily handle a large amount of features. Moreover, under this view, SMT becomes quite similar to sequential natural language annotation problems such as part-of-speech tagging, phrase chunking, and shallow parsing. The training procedure presented in this paper might be used for those problems as well.

The paper is structured as follows: Section 2 introduces the novel local training method by presenting a block orientation bigram model for SMT. Section 3 de-

 $^{^{1}}$ For a more detailed discussion of the block bigram model in comparison to the source-channel approach in [Brown et al. 1993] in conjunction with probabilistic feature weight training as in [Och 2003], see Section 6.

Local Block Orientation

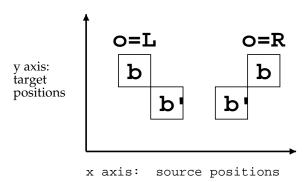


Fig. 2. Block b' is the predecessor of block b. The successor block b occurs with either left o = Lor right o = R orientation. 'left' and 'right' are defined relative to the x axis; 'below' is defined relative to the y axis. For some discussion on global re-ordering see Section 5.3.

scribes details of the localized log-linear prediction model used in this paper. Section 4 describes the online training procedure and compares it to the well known perceptron training algorithm [Collins 2002]. Section 4.1 presents implementation details of the maximum entropy SGD training procedure. Section 5 shows experimental results on an Arabic-English translation task for two block-based translation models: an orientation model and a phrase-distortion model. Section 6 presents a final discussion.

BLOCK ORIENTATION MODEL

This section describes a phrase-based model for SMT similar to the models presented in [Koehn et al. 2003; Och et al. 1999; Tillmann and Xia 2003]. In our paper, phrase pairs are named blocks and our model is designed to generate block sequences. We also model the position of blocks relative to each other: this is called orientation. To define block sequences with orientation, we define the notion of block orientation bigrams. Starting point for collecting these bigrams is a block set $\Gamma = \{b = (S, T) = (s_1^J, t_1^I)\}$. Here, b is a block consisting of a source phrase S and a target phrase T. J is the source phrase length and I is the target phrase length. Single source and target words are denoted by s_i and t_i respectively, where $j=1,\cdots,J$ and $i=1,\cdots,I$. We will also use a special single-word block set $\Gamma_1 \subseteq \Gamma$ which contains only blocks for which J = I = 1. For the experiments in this paper, the block set is the one used in [Al-Onaizan et al. 2004]. Although this is not investigated in the present paper, different block sets may be used for computing the block statistics introduced in this paper, which may affect translation results.

For the block set Γ and a training sentence pair, we carry out a two-dimensional pattern matching algorithm to find adjacent matching blocks along with their position in the coordinate system defined by source and target positions (see Figure 2). Here, we do not insist on a consistent block coverage as one would do during decoding. Among the matching blocks, two blocks b' and b are adjacent if the target phrases T and T' as well as the source phrases S and S' are adjacent. b' is prede-

cessor of block b if b' and b are adjacent and b' occurs below b. A right adjacent successor block b is said to have right orientation o = R. A left adjacent successor block is said to have left orientation o = L. There are matching blocks b that have no predecessor, such a block has neutral orientation (o = N). After matching blocks for a training sentence pair, we look for adjacent block pairs to collect block bigram orientation events e of the type e = (b', o, b). Our model to be presented in Section 3 is used to predict a **future** block orientation pair (b, o) given its predecessor block **history** b'. In Figure 1, the following block orientation bigrams occur: $(\cdot, N, b_1), (b_1, L, b_2), (\cdot, N, b_3), (b_3, R, b_4)$. Collecting orientation bigrams on all parallel sentence pairs, we obtain an orientation bigram list e_1^N :

$$e_{1}^{N} = [e_{1}^{n_{s}}]_{s=1}^{S} = [(b_{i}^{'}, o_{i}, b_{i})_{i=1}^{n_{s}}]_{s=1}^{S}$$
 (2)

Here, n_s is the number of orientation bigrams in the s-th sentence pair. The total number N of orientation bigrams $N = \sum_{s=1}^S n_s$ is about N=7.8 million for our training data consisting of $S=230\,000$ sentence pairs. Note that under different contexts, we have used the symbol N to denote either the total bigram counts or the neutral orientation, which shall not lead to any confusion. In the following, we shall also adopt the convention of using symbol N to indicate other counts. For example, N(b) denotes the counts of block b.

The orientation bigram list is used for the parameter training presented in Section 3. Ignoring the bigrams with neutral orientation N reduces the list defined in Eq. 2 to about 5.0 million orientation bigrams. The **Neutral** orientation is handled separately as described in Section 5.1. Using the reduced orientation bigram list, we collect unigram orientation counts $N_o(b)$: how often a block occurs with a given orientation $o \in \{L, R\}$. $N_L(b) > 0.25 \cdot N_R(b)$ typically holds for blocks b involved in block swapping and the orientation model $p_o(b)$ is defined as:

$$p_o(b) = \frac{N_o(b)}{N_L(b) + N_R(b)}.$$

In order to train a block bigram model as described in Section 3.3, we collect block orientation bigrams (b', o, b) for each training sentence pair $s \in \{1, \dots, S\}$. During the collection, for a given predecessor block b', there might be several successor blocks b. If we were to compute a Viterbi block alignment for a training sentence pair, each block b' in this block alignment would have at most 1 successor: blocks may have several successors, because we do not enforce any kind of consistent coverage during training. All the successor blocks b have to occur in one of the triples in the orientation bigram list e_1^N . The average number of successors for a block b' is 1.5 for the block set used in orientation model experiments in Section 5.1 and 2.0 for the phrase-distortion model experiments in Section 5.3 which use a slightly different block set.

For a block orientation bigram (b', o, b) that occurs in the training data, we call the block b a **'positive'** or 'true' block successor. The training procedure in Section 3 is used to train a log-linear model that discriminates 'positive' block successors against so-called **'negative'** or alternative block successors. The alternative block successors are derived from source and target phrase matches of their corresponding 'positive' block successors. The method is illustrated in Figure 4 and in Figure 5 of Section 5.

LOCALIZED PREDICTION MODEL AND DISCRIMINATIVE TRAINING

In this section, we describe the components used to compute the block bigram probability $p(b_i, o_i | b_{i-1}, o_{i-1})$ in Eq. 1. A block orientation pair (o', b'; o, b) is represented as a feature-vector $f(b, o; b', o') \in \mathbb{R}^d$. For a model that uses all the real-valued components defined below, d is 5. The dimensionality of the feature vector including the binary features depends on the number of binary features. As feature-vector components, we take the negative logarithm of some block model probabilities. We use the term 'float' feature for these feature-vector components (the model score is stored as a float number). Additionally, we use binary block features. The letters (a)-(f) refer to Table III in Section 5.1 which summarizes the feature types used for the orientation model:

Unigram Models: (a),(b) the unigram probability p(b) and (b) the orientation probability $p_o(b)$. These probabilities are simple relative frequency estimates based on unigram and unigram orientation counts derived from the data in Eq. 2. For details see [Tillmann 2004]. During decoding, the unigram probability is normalized by the source phrase length.

Two types of Trigram language model: (c),(d) (c) probability of predicting the first target word in the target clump of b_i given the final two words of the target clump of b_{i-1} , (d) probability of predicting the rest of the words in the target clump of b_i . The language model is trained on a separate corpus.

Lexical Weighting: (e) the lexical weight $p(S \mid T)$ of the block b = (S, T) is computed similarly to [Koehn et al. 2003], details are given in Section 3.1.

Binary features: (f) binary features are defined using an indicator function $f_k(b, b')$ which is 1 if the block pair (b, b') occurs more often than a given threshold C, e.g C = 2. Here, the block orientation o is ignored.

$$f_k(b,b') = \begin{cases} 1 & N(b,b') > C \\ 0 & \text{else} \end{cases}$$
 (3)

3.1 Lexical Weighting Details

The lexical weight $p(S \mid T)$ of the block b = (S, T) is computed similarly to [Koehn et al. 2003], but the lexical translation probability p(s|t) is derived from the block set itself rather than from a word alignment, resulting in a simplified training. The lexical weight is computed as follows:

$$p(S \mid T) = \prod_{j=1}^{J} \frac{1}{N_{\Gamma}(s_j, T)} \sum_{t=1}^{I} p(s_j \mid t_i)$$
$$p(s_j \mid t_i) = \frac{N(b)}{\sum_{b' \in \Gamma_1(b)} N(b')}$$

Here, the single-word-based translation probability $p(s_j \mid t_i)$ is derived from the block set itself. $b = (s_j, t_i)$ and $b' = (s_j, t_k) \in \Gamma_1(b)$ are single-word blocks, where source and target phrases are of length 1. $N_{\Gamma}(s_j, t_1^I)$ is the number of blocks $b_k = (s_j, t_k)$ for $k \in 1, \dots, I$ for which $p(s_j | t_k) > 0.0$.

3.2 Global Model

For the block prediction model, for a given source sentence s, each translation is represented as a sequence of block/orientation pairs $\{b_1^n, o_1^n\}$ consistent with the source language sentence. Using features such as those described above, we can parameterize the probability of such a sequence as $P(b_1^n, o_1^n|w, s)$, where w is a vector of unknown model parameters to be estimated from the training data. We use a log-linear probability model and maximum likelihood training— the parameter w is estimated by maximizing the joint likelihood over all sentences. Denote by $\Delta(s)$ the set of possible block/orientation sequences $\{b_1^n, o_1^n\}$ that are consistent with the source sentence s, then a log-linear probability model can be represented as

$$P(b_1^n, o_1^n | w, s) = \frac{\exp(w^T f(b_1^n, o_1^n))}{Z(s)},$$
(4)

where $f(b_1^n, o_1^n)$ denotes the feature vector of the corresponding block translation, and the partition function is:

$$Z(s) = \sum_{\{b'_1^m, o'_1^m\} \in \Delta(s)} \exp(w^T f({b'_1^m, o'_1^m})).$$

A disadvantage of this approach is that the summation over $\Delta(s)$ can be rather difficult to compute. Consequently some sophisticated approximate inference methods are needed to carry out the computation. A detailed investigation of the global model will be left to another study.

3.3 Local Model Restrictions

In the following, we consider a simplification of the direct global model in Eq. 4. As in [Tillmann 2004], the block bigram probability $p(b_i, o_i \in \{L, R, N\} | b_{i-1}, o_{i-1})$ in Eq. 1 is modeled using two cases separately: (1) $o_i \in \{L, R\}$, and (2) $o_i = N$ (the neutral orientation case where $o_i = N$ is handled as a default case as explained in Section 5.1). Orientation is modeled only in the context of immediate neighbors for blocks that have left or right orientation. The log-linear model is defined as:

$$p(b,o \in \{L,R\} \mid b',o';w,s) \ = \ \frac{\exp(w^T f(b,o;b',o'))}{Z(b',o';s)},$$

where s is the source sentence, f(b, o; b', o') is a locally defined feature vector that depends only on the current and the previous oriented blocks (b, o) and (b', o'). The features were described at the beginning of the section. The partition function is given by

$$Z(b', o'; s) = \sum_{(b,o)\in\Delta(b',o';s)} \exp(w^T f(b,o;b',o')).$$
 (5)

Under this model, the log-probability of a possible translation of a source sentence s, as in Eq. 1, can be written as

$$\ln P(b_1^n, o_1^n | w, s) = \sum_{i=1}^n \ln \frac{\exp(w^T f(b_i, o_i; b_{i-1}, o_{i-1}))}{Z(b_{i-1}, o_{i-1}; s)}.$$
 (6)

The set $\Delta(b', o'; s)$ is a restricted set of possible successor oriented blocks that are consistent with the current block position and the source sentence s, to be

described in the following paragraph. Note that a straightforward normalization over all block orientation pairs in Eq. 5 is not feasible: there are tens of millions of possible successor blocks b (if we do not impose any restriction). For each block b = (S, T), aligned with a source sentence s, we define a source-induced alternative set:

 $\Gamma(b) = \{\text{all blocks } b'' \in \Gamma \text{ that share an identical source phrase with } b\}$

The set $\Gamma(b)$ contains the block b itself and the block target phrases of blocks in that set might differ. To restrict the number of alternatives further, the elements of $\Gamma(b)$ are sorted according to the unigram count N(b'') and we keep at most the top 9 blocks for each source interval s. The partition function is computed slightly differently during training and decoding:

Training:. For each event (b', o, b) that occurs for a sentence pair s in Eq. 2 we compute the successor set $\delta_s(b')$. This defines a set of 'true' block successors. For each true successor b, we compute the alternative set $\Gamma(b)$. $\Delta(b', o'; s)$ is then defined as the union of the alternative sets for each successor b. Here, the orientation o from the true successor b is assigned to each alternative in $\Gamma(b)$. We obtain on the average 12.8 alternatives per training event (b', o, b) in the list e_1^N .

Decoding:. Here, each block b that matches a source interval following b' in the input sentence s is a potential successor. We simply set $\Delta(b',o';s) = \Gamma(b)$, i.e we do not first collect the same alternative set $\Delta(b',o';s)$ used during training which is difficult in the current decoder implementation 2 . An additional simplification speeds up decoding significantly without sacrificing performance: the partition function over the block successors in $\Delta(b',o';s)$ is not actually computed (the decoding speed is improved by more than a factor 10 by this simplification). Simply setting $Z(b',o';s) = w^T \cdot f(b,o;b',o') + 0.5$ does not change performance, i.e. during decoding when predicting a block orientation pair (b,o), the normalization is only approximated by adding 0.5 to the linear product for that block orientation pair. The list $\Gamma(b)$ just restricts the possible target translations for a source phrase. No change in decoding performance is observed due to this approximation and the corresponding results are therefore not reported in Section 5.

In the maximum-likelihood training, we find w by maximizing the sum of the log-likelihood over lists of observed block orientation bigrams as defined in Eq. 2. For a single training sentence pair s that likelihood has the form in Eq. 6. Although the training methodology is similar to the global formulation given in Eq. 4, this localized version is computationally much easier to manage since the summation in the partition function $Z(b_{i-1}, o_{i-1}; s)$ is now over a relatively small set of candidates. This computational advantage is the main reason that we adopt the local model in this paper.

²The block bigram training is implemented separately and as pointed out in Section 2, no consistent block coverage of the training sentences is enforced as one would do during decoding. This leads to a significant training speed-up.

3.4 Global versus Local Models

Both the global and the localized log-linear models described in this section can be considered as maximum-entropy models, similar to those used in natural language processing, e.g. maximum-entropy models for POS tagging and shallow parsing. In the parsing context, global models such as those in Eq. 4 are sometimes referred to as *conditional random fields* or CRF [Lafferty et al. 2001].

Although there are some arguments that indicate that this approach has some advantages over localized models such as Eq. 5, the potential improvements are relatively small, at least in NLP applications. For SMT, the difference can be potentially more significant. This is because in our current localized model, successor blocks of different sizes are directly compared to each other, which is intuitively not the best approach (i.e., probabilities of blocks with identical lengths are more comparable). This issue is closely related to the phenomenon of multiple counting of events, which means that a given source/target sentence pair yields orientation bigrams (b', o, b) where the same block b' preceds different possibly overlapping successor blocks b (with the same or different orientation). In the current training procedure, each successor block b is taken as the truth in turn, without considering the other (possibly also correct) successor blocks that might also occur in that training sentence. In the global modeling, with appropriate normalization, this issue becomes less severe. With this limitation in mind, the localized model proposed here is still an effective approach, as demonstrated by our experiments. Moreover, it is simple both computationally and conceptually. Various issues such as the ones described above can be addressed with more sophisticated modeling techniques, which shall be left to future studies.

4. ONLINE TRAINING ALGORITHM

The local model described in Section 3 leads to the following abstract maximum entropy training formulation:

$$\hat{w} = \arg\min_{w} \sum_{i=1}^{m} \ln \frac{\sum_{j \in \Delta_{i}} \exp(w^{T} x_{i,j})}{\exp(w^{T} x_{i,y_{i}})}.$$
 (7)

In this formulation, w is the weight vector which we want to compute. The set Δ_i consists of candidate labels for the i-th training instance, with the true label $y_i \in \Delta_i$. The labels here are block identities , Δ_i corresponds to the alternative set $\Delta(b',o';s)$ and the 'true' blocks are the 'positive' block successors defined in Section 2. The vector $x_{i,j}$ is the feature vector of the i-th instance, corresponding to label $j \in \Delta_i$. The symbol x is short-hand for the feature-vector f(b,o;b',o'). This formulation is slightly different from the standard maximum entropy formulation typically encountered in NLP applications, in that we restrict the summation over a subset Δ_i of all labels.

Intuitively, this method favors a weight vector such that for each i, $w^T x_{i,y_i} - w^T x_{i,j}$ is large when $j \neq y_i$. This effect is desirable since it tries to separate the correct classification from the incorrect alternatives. If the problem is completely separable, then it can be shown that the computed linear separator, with appropriate regularization, achieves the largest possible separating margin. The effect is similar to some multi-category generalizations of support vector machines (SVM). How-

ever, Eq. 7 is more suitable for non-separable problems (which is often the case for SMT) since it directly models the conditional probability for the candidate labels.

A related method is multi-category perceptron, which explicitly finds a weight vector that separates correct labels from the incorrect ones in a mistake driven fashion [Collins 2002]. The method works by examining one sample at a time, and makes an update $w \to w + (x_{i,y_i} - x_{i,j})$ when $w^T(x_{i,y_i} - x_{i,j})$ is not positive. To compute the update for a training instance i, one usually picks the j such that $w^T(x_{i,y_i} - x_{i,j})$ is the smallest. It can be shown that if there exist weight vectors that separate the correct label y_i from incorrect labels $j \in \Delta_i$ for all $j \neq y_i$, then the perceptron method can find such a separator. However, it is not entirely clear what this method does when the training data are not completely separable. Moreover, the standard mistake bound justification does not apply when we go through the training data more than once, as typically done in practice. In spite of some issues in its justification, the perceptron algorithm is still very attractive due to its simplicity and computational efficiency. It also works quite well for a number of NLP applications.

In the following, we show that a simple and efficient online training procedure can also be developed for the maximum entropy formulation Eq. 7. The proposed update rule is similar to the perceptron method but with a soft mistake-driven update rule, where the influence of each feature is weighted by the significance of its mistake. The method is essentially a version of the so-called stochastic gradient descent method, which has been widely used in complicated stochastic optimization problems such as neural networks. It was argued recently in [Zhang 2004] that this method also works well for standard convex formulations of binary-classification problems including SVM and logistic regression. Convergence bounds similar to perceptron mistake bounds can be developed, although unlike perceptron, the theory justifies the standard practice of going through the training data more than once. In the non-separable case, the method approximately solves a regularized version of Eq. 7, which has the statistical interpretation of estimating the conditional probability. Consequently, it does not have the potential issues of the perceptron method which we pointed out earlier. Due to the nature of online update, just like perceptron, this method is also very simple to implement and is scalable to large problem size. This is important in the SMT application because we can have a huge number of training instances which we may not be able to keep in memory at the same time.

In stochastic gradient descent, we examine one training instance at a time. At the i-th instance, we derive the update rule by maximizing with respect to the term associated with the instance

$$L_i(w) = \ln \frac{\sum_{j \in \Delta_i} \exp(w^T x_{i,j})}{\exp(w^T x_{i,y_i})}$$

in Eq. 7. We do a gradient descent localized to this instance as $w \to w - \eta_i \frac{\partial}{\partial w} L_i(w)$, where $\eta_i > 0$ is a parameter often referred to as the learning rate. For Eq. 7, the update rule becomes:

$$w \to w + \eta_i \frac{\sum_{j \in \Delta_i} \exp(w^T x_{i,j}) (x_{i,y_i} - x_{i,j})}{\sum_{j \in \Delta_i} \exp(w^T x_{i,j})}.$$
 (8)

Similar to online algorithms such as the perceptron, we apply this update rule one by one to each training instance (randomly ordered), and may go through data points repeatedly. Compare Eq. 8 to the perceptron update, there are two main differences, which we discuss below.

The first difference is the weighting scheme. Instead of putting the update weight to a single (most mistaken) feature component, as in the perceptron algorithm, we use a soft-weighting scheme, with each feature component j weighted by a factor $\exp(w^Tx_{i,j})/\sum_{k\in\Delta_i}\exp(w^Tx_{i,k})$. A component j with larger $w^Tx_{i,j}$ gets more weight. This effect is in principle similar to the perceptron update. The smoothing effect in Eq. 8 is useful for non-separable problems since it does not force an update rule that attempts to separate the data. Each feature component gets a weight that is proportional to its conditional probability.

The second difference is the introduction of a learning rate parameter η_i . For the algorithm to converge, one should pick a decreasing learning rate. In practice, however, it is often more convenient to select a fixed $\eta_i = \eta$ for all i. This leads to an algorithm that approximately solves a regularized version of Eq. 7. If we go through the data repeatedly, one may also decrease the fixed learning rate by monitoring the progress made each time we go through the data. For practical purposes, a fixed small η such as $\eta = 10^{-5}$ is usually sufficient. We typically run forty updates over the training data.

A convergence theorem that is similar to the mistake bound analysis for a perceptron algorithm can be proved. For completeness, we include it in Appendix A. Note that an advantage of this method over standard maximum entropy training such as generalized iterative scaling [Darroch and Ratcliff 1972] or improved iterative scaling [Della Pietra et al. 1997] is that it does not require us to store all the data in memory at once. Moreover, the convergence analysis can be used to show that if m is large, we can get a very good approximate solution by going through the data only once. This desirable property implies that the method is particularly suitable for large scale problems. Another advantage of this training algorithm is that it can handle non-binary features easily (while some of the previously proposed methods cannot), which is important for the SMT application we consider here. Although this algorithm is proposed in the context of SMT, it is not difficult to see that it can be used in other areas of NLP with the same advantages mentioned above.

4.1 Implementation Details

In this section some implementation details for the training procedure given in Section 4 are presented. The training procedure is carried out by running over the training data that consists of datapoints that are obtained by collecting block orientation bigrams in the parallel training data. For each block orientation bigram (b', o, b) an alternative set is computed based on 'source' and 'target' phrase alternatives. This is described in more detail in Section 5.1 and in Section 5.3. Even though the phrase-distortion model does not make use of the orientation o of a block b, its training is based on the block orientation bigram list in Eq. 2. Again, successor blocks b with neutral orientation are ignored, i.e. 'positive' blocks are adjacent neighbors with left or right orientation. An example datapoint used in the training data is given in Figure 3. Here, each feature vector f(b', b) is written on a single line. The number 1 at the beginning of a line denotes a feature vector

Table I. Online Training Algorithm: I is the number of iterations, N is the number of data points. The weight update procedure is shown in Table II.

```
initial weight vector w=w_0 for k=1,\cdots,I for i=1,\cdots,N in random order update w on \Delta_i
```

Table II. Update procedure on a datapoint Δ : 'positive' and 'negative' feature vectors are processed differently. T is the sum over the exponential terms, and η is the learning rate.

```
for all 'positive' feature vectors x in \Delta compute t, t_i, T:

t := \exp(w^T x);

T := t + \exp(-4);

for all 'negative' feature vectors x_i

t_i := \exp(w^T x_i);

T + = t_i;

use t, t_i, T to update weight vector w:

\delta = \eta \cdot \left(1.0 - \frac{t}{T}\right);

w \leftarrow w + \delta \cdot x;

for all 'negative' feature vectors x_i

\delta_i = \eta \cdot \left(-\frac{t_i}{T}\right);

w \leftarrow w + \delta_i \cdot x_i;
```

of a 'true' block bigram, the integer -1 denotes an alternative block bigram. A single 0 on a line is used to separate data points. The second integer on every line shows the number of 'float' features followed the actual feature values as well as the number of binary features. Each binary feature is denoted by its index k.

The online training procedure shown in Table 4.1 iterates over the N datapoints in the training data. The training procedure starts with an initial weight vector w_0 where all entries are 0. The weight vector w is updated on each data point using the update procedure in Table II. The update procedure operates on 'positive' and 'negative' feature vectors, e.g. the datapoint in Fig 3 consists of one 'positive' and 5 'negative' feature vectors. For each data point, the outer-most loop is over the 'positive' feature vectors. The sum T of exponential terms $\exp(w^Tx)$ is computed summing over the 'positive' and 'negative' vectors. Here, T is initialized by a small constant $\exp(-4)$, which corresponds to an alternative state that cannot be reached (or a sink state). It is not essential, but only included for stability. In a second loop over the 'positive' and 'negative' feature vectors, the weight vector w is updated by $\eta \cdot (1.0 - \frac{t}{T})$ for the 'positive' feature vector and by $\eta \cdot (-\frac{t}{T})$ for the 'negative' feature vectors. Note that a feature vector component is always non-zero for the

```
1 7 0.235557 0.0715918 1.87109 0 0.971041 1.23743 -1 1 1148003
-1 7 0.125208 0.854685 1.87109 0 2.09924 1.87181 -1 0
-1 7 1.6265 0.0034751 1.87109 0 18 18 -1 0
-1 7 1.01494 0.092259 2.51562 0 0.971041 1.23743 -1 0
-1 7 1.18176 0.092259 2.51562 0.691406 0.971041 1.23743 -2 1 5865644
-1 7 1.06491 0.0715918 1.14453 1.88281 0.971041 1.23743 -2 0
```

Fig. 3. Typical 'datapoint' for the local prediction model. Each block bigram event is represented as a feature vector: all the 'float' features as well as a list of binary features are given. 'Positive' events are marked by 1, negative events are marked by -1 at the beginning of a line.

'float' features, i.e. the 'float' feature weights are updated on every data point. The k-th weight vector component w_k is only updated if the k-th feature is present in the feature vector. Even though it is not necessary for the online training procedure, the training data is first read into main memory in order to avoid costly file access. The SGD training with several million features typically takes about 15 minutes on a single 64-bit AMD opteron 2.4 GHz processor.

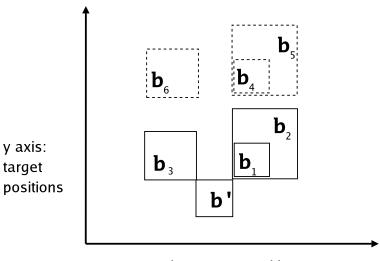
5. EXPERIMENTAL RESULTS

The log-linear block bigram approach 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 [Papineni et al. 2002] are presented on the MT03 Arabic-English DARPA evaluation test set consisting of 663 sentences with 16 278 Arabic words with 4 reference translations. For some of the experiments in Section 5.1 the so-called LDC devtest set is used. The data is the development data provided by LDC for the 2002 DARPA Arabic-English MT evaluation. It consists of 1043 sentences with 25 889 Arabic words with 4 reference translations.

In order to speed up the parameter training the original training data is filtered according to the test set: all the Arabic substrings up to length 12 that occur in the test set are computed and the parallel training data is filtered to include only those training sentence pairs that contain at least one out of these phrases: the resulting 'MT03' training data contains about 230 000 sentence pairs, while the 'LDC' training data contains about 273 000 sentence pairs. Additionally, for the experiments in Section 5.1, the block set is also pre-filtered to include only those blocks whose source phrase matches some source phrase in the test set. The size of the resulting block set is about 242 000 blocks for 'MT03' training data, and about 281 000 blocks for the 'LDC' training data.

A different block set is used for the experiments in Section 5.3: this block set is not pre-filtered according to a particular test data and consists of about 9 532 000 blocks. All three block sets are derived using a phrase-pair selection algorithm

Source Interval Alternatives



x axis: source positions

Fig. 4. For each of the blocks b_1 , b_2 , and b_3 , a source-match derived alternative block is shown that covers the same source interval (b_4 is an alternative for b_1 , b_5 is an alternative for b_2 , etc.).

similar to [Koehn et al. 2003; Tillmann and Xia 2003; Tillmann 2003]. 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 [Al-Onaizan et al. 2004]. In all the experiments reported, word casing is added as a post-processing step using a statistical model (details are omitted here) ³.

5.1 Orientation Model

For the block orientation model, model performance is analyzed with respect to the number and type of features used as well as with respect to different re-ordering models. During decoding, a list of block orientation bigrams is generated as described in Section 2. A DP-based beam search procedure identical to the one used in [Tillmann 2004] is used to maximize the block orientation probability $P(b_1^n, o_1^n)$ in Eq. 1 over all oriented block segmentations (b_1^n, o_1^n) . Here, a simple heuristic is used to speed up decoding, which typically does not reduce translation performance in terms of BLEU: bigrams (b', L, b) with left orientation are only generated if $N_L(b) \geq 3$ for the successor block b. The generation of 'negative' block alternatives is demonstrated in Figure 4. Here, the block orientation bigrams (b', R, b_1) , (b', R, b_2) , and (b', L, b_3) occur in the training data. Alternatives are derived from source phrase matches, e.g. the block b_4 is an alternative for block b_1 , since both cover the same source interval.

Results for 9 experiments are shown in Table IV, where the feature types are described in Table III. The first 5 experimental results are obtained by carrying

 $^{^3}$ Translation performance is typically improved by about 2.5 % ignoring the case information. ACM Journal Name, Vol. V, No. N, M 2007.

Table III. List of feature-vector components used for the orientation model results in Section 5.1. For a description, see the beginning of Section 3.

Description

- (a) Unigram probability
- (b) Orientation probability
- (c) LM first word probability
- (d) LM second and following words probability
- (e) Lexical weighting
- (f) Binary Features

out the likelihood training described in Section 3. Line 1 in Table IV shows the performance of the baseline block unigram 'MON' model which uses two 'float' features: the unigram probability and the boundary-word language model probability. No block re-ordering is allowed for the baseline model (a **monotone** block sequence is generated). The 'SWAP' model in line 2 uses the same two features, but neighbor blocks can be swapped. No performance increase is obtained for this model. The 'SWAP & OR' model uses an orientation model as described in Section 2 and Figure 2. Here, we obtain a small but significant improvement over the baseline model ⁴. Line 4 shows that by including two additional 'float' features: the lexical weighting and the language model probability of predicting the second and subsequent words of the target clump yields a further significant improvement. Line 5 shows that including binary features and training their weights on the training data actually decreases performance. This issue is addressed below. The model training is carried out as follows: the results in line 1-4 are obtained by training 'float' weights only. Here, the training is carried out by running only once over 10 % of the training data. The model including the binary features is trained on the entire training data. About 3.37 million features of the type defined in Eq. 3 are obtained by setting a cut-off threshold of C=3. Forty iterations over the training data take a few minutes on a single Intel machine. Although the online algorithm does not require us to do so, our training procedure keeps the entire training data and the weight vector w in about 2 gigabytes of main memory.

For blocks with neutral orientation o = N, a separate model is trained that does not use the orientation model feature. E.g. for the results in line 5 in Table IV, the neutral model would use the features (a), (c), (d), (e), but not (b) and (f). **BB**-type binary features are not included into neutral orientation model for a successor block b, since such a block does not have an immediate predecessor block b' below itself. The neutral model is trained on the neutral orientation bigram subsequence that is part of Eq. 2. During decoding the decoder keeps track of the orientation of each newly generated block. This can be done locally without having to wait for the entire block translation to be finished. The decoder maintains two weight vectors: one for left and right orientation blocks (a 6-dimensional vector for the model in line 5 in Table IV) and one for neutral orientation blocks (a 4-dimensional vector): depending on its orientation, the probability of the successor block is computed using the appropriate weight vector and its corresponding components.

⁴The results in the first three lines of Table IV can be compared to the results presented in [Tillmann 2004], where identical feature components were used, but the feature weights were set

Table IV. Orientation model translation results in terms of cased BLEU with confidence intervals on the MT03 test data. The second column shows the local re-ordering model, the third column summarizes the model variations. The results in lines 8 and 9 are for a cheating experiment: the float weights are trained on the test data itself.

	Re-ordering	Components	BLEU
1	'MON'	(a),(c)	32.3 ± 1.5
2	'SWAP'	(a),(c)	32.3 ± 1.5
3	'SWAP & OR'	(a),(b),(c)	33.9 ± 1.4
4	'SWAP & OR'	(a)-(e)	37.7 ± 1.5
5	'SWAP & OR'	(a)-(f)	37.2 ± 1.6
6	'SWAP & OR'	(a)-(e) (ldc devtest)	37.8 ± 1.5
7	'SWAP & OR'	(a)-(f) (ldc devtest)	38.2 ± 1.5
8	'SWAP & OR'	(a)-(e) (mt03 test)	39.0 ± 1.5
9	'SWAP & OR'	(a)- (f) $(mt03 test)$	39.3 ± 1.6

5.2 Modified Training for Orientation Model

The following variation of the likelihood training procedure described in Section 3 is implemented, where the 'LDC' devtest set is used. First, a model is trained on the 'LDC' training data using 5 float features and the **BB**-type binary features. This model is used to decode the devtest 'LDC' set. During decoding, a 'translation graph' is generated for every input sentence using a procedure similar to [Ueffing et al. 2002]: a translation graph is a compact way of representing candidate translations which are close in terms of likelihood. From the translation graph, the 1000 best translations according to the translation score are obtained. From this list, the block sequence that generated the top BLEU-scoring target translation is computed. Computing the top BLEU-scoring block sequence for all the input sentences, the following list is generated:

$$e_1^{N'} = [(b_i', o_i, b_i)_{i=1}^{n_{s'}}]_1^{S'},$$
 (9)

where $N' \approx 9400$. Here, N' is the number of blocks needed to decode the entire devtest set. Alternatives for each of the events in $e_1^{N'}$ are generated using the 'source' alternative method illustrated in Figure 4. This set of block alternatives is further restricted by using only those blocks that occur in some translation in the 1000-best list. The 5 float weights are trained on the modified training data in Eq. 9, where the training takes only a few seconds. The 'MT03' test set is decoded using the modified 'float' weights. As shown in line 4 and line 6 of Table IV, there is almost no change in performance between training on the original training data in Eq. 2 or on the modified training data in Eq. 9. Line 8 shows that even when training the float weights on an event set obtained from the test data itself in a cheating experiment, only a moderate performance improvement from 37.7 to 39.0 is obtained. For the experimental results in line 7 and 9, the same five float weights as trained for the experiments in line 6 and 8 are used and kept fixed while training the binary feature weights only. Using the BB-type binary features leads to only a minor improvement in BLEU performance from 37.8 to 38.2 in line 7. For this best model, a 18.6 % BLEU improvement over the baseline is obtained.

by exhaustively searching the weight space.

ACM Journal Name, Vol. V, No. N, M 2007.

Table V. List of feature-vector components used for the phrase distortion model results in Section 5.3. For a description, see Section 5.3.

8 0.0	
	Description
	(a) Direct translation probability
	(b) Lexical weighting
	(c) LM first word probability
	(d) LM second and following words probability
	(e) Distortion probability 'inbound'
	(f) Distortion probability 'outbound'
	(g) Negative target phrase length
	(h) Binary Features

5.3 Phrase Distortion Model

In order to further validate the novel training technique which is based on enumerating local block neighbors, a set of experiments is carried out using a modified block-based translation model. Again, a block pair (b_i, b_{i-1}) is represented as a feature vector $f(b, b') \in \mathbb{R}^d$, which includes a set of 7 real-valued features. The block orientation component is replaced by a phrase-distortion model. Additionally, the unigram probability is replaced by the direct translation probability, and a brevity penalty is introduced. These changes lead to an improved translation performance. In detail the following feature vector components are used which are summarized in Table V:

- **—Direct Model: (a)** 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 phrase T.
- —**Lexical Weighting:** (b) the lexical weight $p(S \mid T)$ of block b = (S, T) is computed similarly to [Koehn et al. 2003]. The lexical weighting depends on the so-called Model 1 translation model probability [Brown et al. 1993]. This model is trained in a separate training step.
- —**Trigram language model:** (c)-(d) The probability of predicting the first target word in the target clump of b_i given the final two words of the target clump of b_{i-1} , and the probability of predicting the rest of the target words in the target clump of b_i .
- **Distortion Weighting:** (e)-(f) A block re-ordering model with distortion probabilities based on source phrase words is used. The distortion weighting model assigns a score to a block b based on its source position relative to its predecessor block b'. The phrase distortion model is computed from word-aligned training data. For details, see [Al-Onaizan et al. 2004].
- —**Target Phrase Length: (g)** The negative number of words in the target phrase of a block. This feature is used to control the total number of words in the final translation.
- —Binary Features: (h) In addition to the block bigram features (BB features) in Section 3, the following features are used:

Target phrase feature (TP feature):

$$f_k(b,b') = \begin{cases} 1 & b = (s,t) \text{ and } b' = (s',t') \text{ and } N(t,t') > N \\ 0 & \text{else} \end{cases}$$
 (10)

Word boundary (WB feature) feature:

$$f_k(b,b') = \begin{cases} 1 & b = (s,t) \text{ and } b' = (s',t') \text{ and } N(t_1,t'_n) > N \\ 0 & \text{else} \end{cases}$$
, (11)

where t_1 is the first word of successor target phrase t and t'_n is the last word of the predecessor target phrase t', i.e. this feature is used to capture typically word bigrams that occur at target phrase boundaries.

The block orientation model as expressed in Eq. 1 is modified to:

$$P(b_1^n) \approx \prod_{i=1}^n p(b_i|b_{i-1}),$$
 (12)

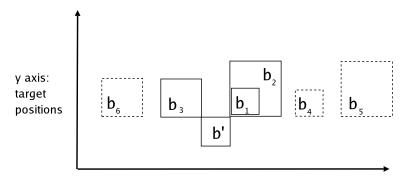
i.e. the block orientation o_i is omitted. A block sequence b_i^n is generated, where local phrase re-ordering is restricted appropriately. For the final block translation, each source and target position is covered by at most 1 block as shown in Figure 1. For simplicity reasons, this coverage restriction is not expressed in our notation. The concept of skipping single-word source positions introduced in [Tillmann and Ney 2003] is extended to handle source phrases (for a detailed description of the phrase-based decoder see [Tillmann 2006]). At most 2 source intervals may be skipped in a left to right traversal of the input sentence, i.e. at most 2 'phrase holes' in the left-to-right traversal of the input sentence are allowed. Additionally, all phrase re-ordering is restricted to take place within a window of size 6 5 . The coverage and cardinality decoder pruning parameters introduced in [Tillmann and Ney 2003] are used and set to conservatively large values, such that the effect of search errors during decoding are negligible.

In a first set of experiments, the orientation model training procedure described in Section 2 and Section 5.1 is applied to train the parameters of the modified phrase-distortion model, i.e. the 'alternative' block generation based on source phrase matches is used. A BLEU score of 26.0 is obtained as shown in line 1 of Table VI. In informal experiments, a simple heuristic has been found that improves translation performance significantly: the weight components w_i of the weight vector w are constrained to be negative for the 'float' feature components, i.e. $w_i < 0$ for $i \in \{1, \dots, 7\}$. The translation performance is improved significantly to 35.0 as shown in line 2 of Table VI. This restriction can be justified as follows: since all the 'float' features are defined as negative logarithms of some component model probabilities, these model 'scores' have to be dealt with homogeneously, avoiding differing signs for their weights. The sign restriction can be readily included in the update algorithm in Table II: for the update step $w \leftarrow w + \delta \cdot x$, if $w_i < 0$ then set $w_i = 0$.

Performance is further improved by the following considerations. Using the source-phrase based alternative generation method, the following problem is encountered: for an adjacent block bigram pair (b', b), the block alternatives b'' of the 'positive' successor block b, cover the same source interval as that block, i.e. the

⁵Further increasing the number of allowable skips or increasing the window size actually decreases performance. To handle global re-ordering, the block-based translation model would need to be extended, e.g. by including hierarchical blocks [Chiang 2005].

Target Interval Alternatives



x axis: source positions

Fig. 5. Block b' has three successor blocks b_1 , b_2 , and b_3 . Alternatives are computed from source and target phrase matches. Here, only the target-phrase derived alternatives are shown, e.g. block b_4 is an alternative for block b_1 .

feature vector f(b', b) for the 'positive' successor and the feature vector f(b', b'') for the 'negative' successor have identical values for the distortion model components (e) and (f) (5-th and 6-th vector component), since they cover the same source interval. In order for the distortion feature to discriminate between 'positive' and 'negative' block successors, additional block alternatives are introduced based on target phrase matches as illustrated in Figure 5. For each block successor b, a set of blocks b_i is computed that match the same target interval as block b. Here, only blocks b_i are used that actually occur in the parallel training sentence pair and therefore cover some source interval in that sentence pair. These alternatives do not have to be adjacent to b', the phrase distortion model is used to assign distortion cost for any given block pair (b', b). The newly generated alternative blocks are included into a joint alternative set. The effect of the modified alternative set is shown in line 4 of Table VI: BLEU performance is increased to 39.9. Using the negative 'float' weight restriction does not change the performance (the unrestricted training already produced all negative 'float' weights).

The overall training data consists of 6.3 million data points and training the exponential model with up to 15 million features takes only about 15 minutes on a single 64-bit AMD opteron 2.4 GHz processor ⁶. To keep the entire training data in main memory as is needed by the current implementation, only every second data point is used: the total number of datapoints is reduced to 3.15 million. The number of positive block successors per datapoint is 1.98 and the number of alternatives for the source & target alternative generation case is 7.45. Here, the 'positive' block successors are excluded from the alternatives of each data point. For the 'float' feature model only 7 weights have to be trained and the training is carried out on

⁶This number differs from the 5 million block orientation bigrams reported in Section 2. Here, a different block set is used.

Table VI. Phrase-distortion model translation results in terms of cased BLEU with confidence intervals on the MT03 test data. The first column shows the block alternative generation methods: 'source' and 'source & target'. The second column shows 'float' weights training restrictions, and the third column reports the binary features types that have been used.

	Alternatives	'float' restrictions	feature type	BLEU
1	Source	unconstrained	none	26.1 ± 1.2
2		negative	none	35.0 ± 1.4
3		negative	$\mathbf{B}\mathbf{B}$	35.0 ± 1.4
4	Source & Target	unconstrained	none	39.7 ± 1.6
5		negative	none	39.9 ± 1.6
6		negative	WB	38.3 ± 1.6
7		negative	$_{ m BB}$	39.9 ± 1.6
8		negative	\mathbf{TP}	40.0 ± 1.6

Table VII. Block prediction error rate ϵ_P and cased BLEU performance as a function of the number n_f of 'float' features and the type of binary features used.

J			· · · · · · · · · · · · · · · · · · ·		-
	n_f	binary type	error rate ϵ_P	BLEU	
	1	none	0.5518	22.1 ± 1.1	
	2	none	0.5460	21.9 ± 1.1	
	3	none	0.4713	31.3 ± 1.5	
	4	none	0.4372	34.4 ± 1.4	
	5	none	0.2905	39.5 ± 1.6	
	6	none	0.2905	39.5 ± 1.6	
	7	none	0.2897	39.9 ± 1.6	
	7	WB	0.2524	38.3 ± 1.6	
	7	$_{ m BB}$	0.2638	39.9 ± 1.6	
	7	\mathbf{TP}	0.2412	40.0 ± 1.6	

only 1 % of the overall training data by using only every 100-th datapoint from the overall training data. On these 70 000 data points the weight training takes only a few seconds. Additionally, 230 000 data points have been set aside as test data for the block prediction experiments reported below.

An additional analysis of the discriminative training procedure is presented in Table VII. In this table, the translation as well as the prediction performance for a series of models is reported: a model using all the 7 'float' features listed in Table V achieves a BLEU score of 39.9 %. The prediction performance is measured as the model's capability to distinguish between 'positive' and 'negative' block successors. The prediction error rate ϵ_P on some data is defined as:

$$\epsilon_P = \frac{\text{number of datapoints such that a 'negative' block } b \text{ maximizes } w^T x}{\text{total number of datapoints}}$$
 (13)

Here, w is the weight vector obtained during training and x is the feature vector associated with a block bigram (b',b). In this paragraph, the following intuition is tested: a model that learns to more reliably choose the right block successors also leads to improved block translation performance. To this end, restricted models are defined by reducing the number of 'float' features used: only the first n_f float features are used e.g. for $n_f=3$ only the 'float' features (a)-(c) from Table VII are used. The table shows a clear dependency between block prediction error rate and translation performance. The performance is improved from using only 1 'float' feature with a BLEU score of 22.1 % to a BLEU score of 39.9 using 7 float features.

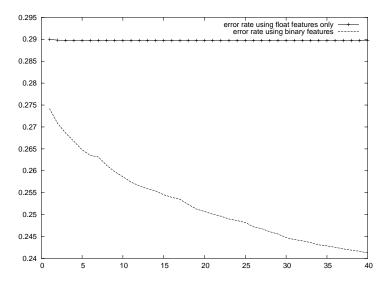


Fig. 6. Block prediction error rate results on the test data consisting of 230 000 datapoints. After each iteration k ($k = 1, \dots, 40$) of the algorithm in Table II the latest weight vector w_k is used to compute the error rate ϵ_P in Eq. 13. The upper graph shows the error rate for the model training that is based on 7 'float' features, the lower graph shows the error rate for a model that additionally uses **TP**-type binary features.

The error rate ϵ_P is reduced accordingly: the more 'float' features are used, the more reliably the block bigram model predicts the block successor b. In spite of the fact that the binary features reduce the block prediction error rate further, performance is improved only slightly from 39.9 to 40.0 when using the **TP**-type binary features. In the case of the **WB** features performance is even decreased significantly.

Figure 6 reports for each training iteration k of the algorithm in Table II the prediction error rate ϵ_P defined in Eq. 13. The prediction error rate is computed on the test data consisting of 230 000 datapoints. After each iteration k the current weight vector w_k is used to compute this error rate on the test data. The graphs show the prediction error as a function of the training iteration k. The upper graph is for a model based on 7 'float' features (corresponding to line 5 in Table IV) where the training is carried out on only 1 \% of the total training data. Even after the first iteration (starting with an all-zero weight vector) an error rate of 0.2900 is achieved. After the third iteration the error rate drops to 0.2897 and remains constant for all subsequent iterations. The block prediction error rate is also computed for the model corresponding to the last line of Table VII using the **TP**-type binary features. Here, the test set error rate consistently drops from 0.271 after the first iteration to 0.243 after the 40-th iteration. Unfortunately, this improvement in prediction error does currently not result in an improved translation performance. These results can be explained as follows: 1) a low-dimensional weight vector w can be trained by running only once over the training data as mentioned in Section 4. 2) using possibly millions of binary features does not improve performance: one possible explanation is that the binary features make the model overfit the training

'Float' Model 'Float' & 'Binary' Model york 000 vork 0000 new new 0 \cap in in council council security 0security 00 the • 0 0 the 00 in 0000 in m Α f f m A f 1 y y 1 y Α A m y m n w n w r r k k

Fig. 7. Block translation fragment that demonstrates the effect of including the **TP**-type binary features on top of the phrase distortion model. The model including the binary features typically produces the same or a very similar target translation using an increased number of blocks.

data. This problem is compounded by the overlapping block problem described in the following section. For a more detailed analysis of the results, see the following Section 5.4 as well as Section 6.

5.4 Binary Feature Analysis

The current maximum entropy training fails to significantly improve translation performance by including binary features. This fact is more closely examined in this section. As has been demonstrated in the preceding section, the binary features clearly improve prediction performance in terms of the block prediction error rate ϵ_P , but fail to improve translation performance in terms of BLEU error rate. A simple analysis is carried out by looking at the actual decoder output ⁷. Even though translation performance seems virtually unchanged in terms of BLEU score, almost half of the test sentences are translated slightly differently using the model that includes the best-performing **TP**-type binary features. A typical situation is shown in Figure 7. Here, a block translation fragment for the phrase-distortion model (cf. line 5 in Table VI) is shown on the left-hand side. The translation of the Arabic input fragment is carried out using 2 blocks. On the right-hand side, the translation of the same input fragment using a system including **TP**-type binary features is shown (cf. line 8 in Table VI). The same target translation is produced using 4 blocks: the use of the binary features leads to modified block translation without modifying the actual target translation in many cases. Using the binary features, there is a tendency to use an increased number of blocks to produce

 $^{^{7}}$ Even though we actually look at the test data all model components remain unchanged, i.e. the block set and the feature vector.

ACM Journal Name, Vol. V, No. N, M 2007.

the same or a very similar target translation. This observation suggests that the overall effect of using binary features overlaps with the use of large blocks by the phrase-distortion translation model without binary features: the model including binary features learns to predict a block b given its predecessor block b' even if the composite block (b',b) itself is present in the block set used during translation. A possible solution to the problem of overlaping blocks is to train the block orientation model in a global fashion as shown in [Tillmann and Zhang 2006].

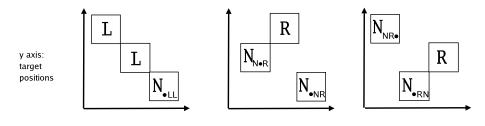
6. DISCUSSION AND FUTURE WORK

As noted in the introduction, statistical machine translation was invented by the IBM Candide machine translation project and their noisy channel approach. The noisy channel approach proposes two probability components: when translating a French sentence f into an English sentence e, rather than modeling the conditional probability $Pr(\mathbf{e}|\mathbf{f})$ directly, the translation process is decomposed into two components: the translation channel probability $Pr(\mathbf{f}|\mathbf{e})$ and the language model probability $Pr(\mathbf{e})$, which are modeled separately. The two components work together: the translation model component produces possibly ill-formed French translations whose word order is controlled by the language model $Pr(\mathbf{e})$. For the more complex models in [Brown et al. 1993], in order to make the training computationally feasible the translation model $Pr(\mathbf{f}|\mathbf{e})$ might even be 'deficient': probability mass is wasted on things that are not even strings, i.e. placing different target words on the same target sentence position. On the contrary, the current block bigram approach can be more easily understood as a direct translation model which models the translation probability $Pr(\mathbf{e}|\mathbf{f})$ directly. The block sequence generation model is closely linked to the actual phrase-based decoder that proceeds from left-to-right over the input sentence (for details on the phrase-based decoder used in this paper, see [Tillmann 2006]). For the block-based model, deficiency is not an issue as the block decoder never generates impossible block orientation sequences: the phrase reodering is restricted to local block swapping only. For more complex re-ordering schemata as in [Brown et al. 1993], a localized training that relies on enumerating local neighbors as described in this paper might not be feasible. Furthermore, the probabilistic model in [Brown et al. 1993] is trained in a global way: all model parameters are trained jointly to optimize the likehood of the training data using the EM algorithm. On the other hand the current paper demonstrates that good performance can be obtained by a purely **local** training that looks only at adjacent block pairs without taking into account the sentential context.

State-of-the-art phrase-based MT models are typically presented as extensions of the source-channel model in [Brown et al. 1993]. In addition to the features used in that paper, i.e. translation, language model, and distortion probabilities, additional features are introduced which might not be based on a probabilistic model (such as features of type g) in Table V). Typically each feature is assigned a weight which expresses its relevance in a log-linear combination of features. The weights can be trained using the minimum-error-rate training presented in [Och 2003]. The current training algorithm differs from that influental algorithm as follows:

(1) For the exponential model presented in this paper, there is no need to decode the training data or to compute N-best lists as is the case for minimum-error-rate

Block Trigram Orientation



x axis: source positions

Fig. 8. Examples of more complex block orientation re-ordering schemata: blocks with neutral orientation are distinguished as to where they occur in the local block sequences. A 'dot' notation is used to denote more complex orientation schemata.

training. The additional decoding step is computational expensive. On the other hand, the current block bigram approach needs to collect local block neighbors in a way that might depend on the model being trained, e.g. the need for model specific training adaptions might arise, e.g. the target phrase induced alternatives necessary for the training in Section 5.3.

(2) The minimum-error-rate training is a **global** training procedure. It trains a weight vector w by minimizing a global error which is defined over all sentences in the training data (typically a development set with multiple reference translations). By updating the weight vector w component-wise carrying out a one-dimensional line search for each dimension of the weight vector, the algorithm cannot handle a high-dimensional feature space. Moreover, even for a low dimensional weight vector the training becomes quite slow, training may take more than an hour (assuming the generation of an 200-best list and 5 to 7 training iterations as reported in [Och 2003]) vs a few seconds or minutes for the method presented in this paper.

Future work might also explore different block orientation schemata. For example Figure 8 shows a slightly more complex block orientation scheme where the block re-ordering is less restrictive: three rather than two neighbor blocks are involved in the re-ordering. The neutral orientation blocks are further distinguished according to their position relative to their immediate neighbor blocks, e.g. in the left-most example the type of neutral orientation assigned to the right-most block also captures the fact that it is followed by two left orientation blocks. The *dot* notation is used to indicate the position of the neutral block within the local orientation sequence, e.g. neutral blocks occur in zero-th position in the first example, in zero-th and first position in the second, and in zero-th and second position for the third example. Other block orientation variants might be investigated in future experiments, e.g. replacing the neutral orienation by additional left and right orientation tags for cases where blocks are placed to the left or the right of the current block, but are not adjacent to it (see [Nagata et al. 2006] for a related model). The general advantage of restricted re-ordering schemata is the following: 1) the phrase-based

machine translation model is 'linearized' and fast DP-based decoding techniques can be used, 2) discriminative training techniques which have been developed for other sequential NLP problems like part-of-speech tagging are applicable (see also [Tillmann and Zhang 2006] for another discriminative training procedure based on block orientation sequences).

As has been shown in Section 5.1 and Section 5.3, the translation performance is largely dominated by the 'float' features. Moreover, the results in Section 5.1 show that performance doesn't change much when training on training, devtest, or even test data. This suggests that the weights for the real-valued features are largely determined by the feature type and not by the choice of the training data. The maximum entropy training method presented here is very efficient and fast, training the 7 real-valued feature model in only about 1 minute on a single machine which is much faster than the error driven training in [Och 2003] with only a minor degradation in performance ⁸.

Even though currently we do not obtain a significant improvement from the use of binary features, we expect their use to be a promising approach in future experiments. A weak point of the current local prediction model is that it does not take into account the block interaction on the sentence level, i.e. the block successors b for a predecessor block b' cover different-size fragments of the input sentence. A more accurate approximation of the global model as discussed in Section 3.2 might improve performance. Section 5.1 and Section 5.3 present two efficient ways of generating local alternatives based on source phrase and target phrase matches. Another approach might be to generate 'global' alternatives that represent an entire translation, i.e a block sequence as a single feature vector and discriminate translation alternatives on the sentence level rather than the block neighbor level. In the context of statistical parsing, similar work on online training algorithms has been presented in [McDonald et al. 2005]. This involves the computationally expensive generation of the best-scoring block sequence for each training sentence using the decoder (potentially each sentence has to be decoded several times). A restricted number of alternatives can be computed by generating a translation graph and extracting a N-best list. Due to the much increased computational requirements this is future work.

As mentioned in Section 1, viewing the translation process as a sequence of local discussions makes it similar to other NLP problems such as POS tagging, phrase chunking, and also statistical parsing. This similarity may facilitate the incorporation of these approaches into our translation model. Discriminative training techniques have been recently applied to statistical machine translation [Moore 2005; Taskar et al. 2005], but only in the context of word-alignment, instead of phrase-based translation itself. As far as the log-linear combination of float features is concerned, similar training procedures have been proposed in [Och 2003]. This paper reports the use of 8 features whose parameters are trained to optimize performance in terms of different evaluation criteria, e.g. BLEU. On the contrary,

⁸The best translation result for a real-valued features only model in Table 6 is 39.9. The same model using the same block set yields a BLEU score of 41.5 when trained using an error-driven training algorithm similar to [Och 2003]. The error-driven training is carried out on a fraction of the MT03 test data itself.

our paper shows that an almost identical performance can be obtained using a likelihood training criterion. Our discriminative training procedure is related to the re-ranking procedure presented in [Shen et al. 2004]. In fact, one may view discriminative reranking as a simplification of the global model we discussed, in that it restricts the number of candidate global translations to make the computation more manageable. However, the number of possible translations is often exponential in the sentence length, while the number of candidates in a typically reranking approach is fixed. Unless one employs an elaborate procedure, the candidate translations may all be very similar to one another, and thus do not give a good coverage of representative translations. Therefore the reranking approach may have some severe limitations which need to be addressed. For this reason, we think that a more principled treatment of global modeling can potentially lead to further performance improvements. For future work, our training technique may be used to train models that handle global sentence-level reorderings. This might be achieved by introducing orientation sequences over phrase types that have been used in ([Schafer and Yarowsky 2003]). To incorporate syntactic knowledge into the block-based model, the use of additional real-valued or binary features will be examined, e.g. features that look at whether the block phrases cross syntactic boundaries. This can be done with only minor modifications to the current training method. In the current phrase-based machine translation approaches [Chiang et al. 2005; Kumar and Byrne 2005] that use syntactic parsing information or explicitly model local re-ordering, some version of the local training method can be applied. For example, in the hierarchical model presented in [Chiang et al. 2005], rule probabilities can be trained by enumerating all the local derivations for a given non-terminal in the grammar.

APPENDIX

A. CONVERGENCE OF STOCHASTIC GRADIENT DESCENT

In this appendix, we present a convergence analysis of the online maximum entropy training algorithm in Section 4. Unlike the traditional convergence analysis for stochastic gradient descent (for example, see [Kushner and Yin 1997]), which does not lead to explicit bounds, the technique we employ here is related to the mistake bound analysis for perceptron (also see [Zhang 2004]). In this setting the convergence bound can be explicitly stated. In particular, using a fixed learning rate, we can obtain the following convergence theorem.

THEOREM A.1. Assume that we apply the update rule Eq. 8 over training points from i = 1, ..., m, starting with an initial weight vector w_0 . Let w_i be the weight vector after seeing the i-th datum, and

$$L_i(w) = \ln \frac{\sum_{j \in \Delta_i} \exp(w^T x_{i,j})}{\exp(w^T x_{i,y_i})}.$$

Then we have the following bound:

$$\frac{1}{m} \sum_{i=1}^{m} L_i(\hat{w}_{i-1}) \le \inf_{\bar{w}} \left[\frac{1}{m} \sum_{i=1}^{N} L_i(\bar{w}) + \frac{\|\bar{w} - w_0\|_2^2}{2\eta m} \right] + \frac{\eta \sup_{i,j} \|x_{i,y_i} - x_{i,j}\|_2^2}{2}.$$

PROOF. Consider step i. Let $p_{i,k} = \exp(\hat{w}_{i-1}^T x_{i,k}) / \sum_{j \in \Delta_i} \exp(\hat{w}_{i-1}^T x_{i,j})$. We consider the following decomposition:

$$\begin{aligned} \|\hat{w}_{i} - \bar{w}\|_{2}^{2} &= \left\| (\hat{w}_{i-1} - \bar{w}) + \eta \sum_{j \in \Delta_{i}} p_{i,j}(x_{i,y_{i}} - x_{i,j}) \right\|_{2}^{2} \\ &= \left\| \hat{w}_{i-1} - \bar{w} \right\|_{2}^{2} + \eta^{2} \left\| \sum_{j \in \Delta_{i}} p_{i,j}(x_{i,y_{i}} - x_{i,j}) \right\|_{2}^{2} \\ &+ 2\eta \sum_{j \in \Delta_{i}} p_{i,j}(x_{i,y_{i}} - x_{i,j})^{T} (\hat{w}_{i-1} - \bar{w}) \end{aligned}$$

$$= \left\| \hat{w}_{i-1} - \bar{w} \right\|_{2}^{2} + \eta^{2} \left\| \sum_{j \in \Delta_{i}} p_{i,j}(x_{i,y_{i}} - x_{i,j}) \right\|_{2}^{2} - 2\eta (L_{i}(\hat{w}_{i-1}) - L_{i}(\bar{w})) + 2\eta \left(L_{i}(\hat{w}_{i-1}) - L_{i}(\bar{w}) + \sum_{j \in \Delta_{i}} p_{i,j}(x_{i,y_{i}} - x_{i,j})^{T} (\hat{w}_{i-1} - \bar{w}) \right) + 2\eta \left(L_{i}(\hat{w}_{i-1}) - L_{i}(\bar{w}) + \sum_{j \in \Delta_{i}} p_{i,j}(x_{i,y_{i}} - x_{i,j}) \right\|_{2}^{2} - 2\eta (L_{i}(\hat{w}_{i-1}) - L_{i}(\bar{w})) - 2\eta \left(L_{i}(\bar{w}) - L_{i}(\hat{w}_{i-1}) - (\bar{w} - \hat{w}_{i-1})^{T} \frac{\partial}{\partial \hat{w}_{i-1}} L_{i}(\hat{w}_{i-1}) \right). \end{aligned}$$

Note that it is well-known that for a convex function $L_i(w)$ of w, the inequality

$$L_i(\bar{w}) - L_i(\hat{w}_{i-1}) - (\bar{w} - \hat{w}_{i-1})^T \frac{\partial}{\partial \hat{w}_{i-1}} L_i(\hat{w}_{i-1}) \ge 0$$

holds (this is easy to verify by definition). Therefore we obtain

$$\|\hat{w}_i - \bar{w}\|_2^2 \le \|\hat{w}_{i-1} - \bar{w}\|_2^2 + \eta^2 \left\| \sum_{j \in \Delta_i} p_{i,j} (x_{i,y_i} - x_{i,j}) \right\|_2^2 - 2\eta (L_i(\hat{w}_{i-1}) - L_i(\bar{w})).$$

Summing over i = 1, ..., m, we obtain

$$\|\hat{w}_m - \bar{w}\|_2^2 \le \|\hat{w}_0 - \bar{w}\|_2^2 + \eta^2 \sum_{i=1}^m \left\| \sum_{j \in \Delta_i} p_{i,j} (x_{i,y_i} - x_{i,j}) \right\|_2^2 - 2\eta \sum_{i=1}^m (L_i(\hat{w}_{i-1}) - L_i(\bar{w})).$$

By rearranging this inequality, we obtain the desired bound. \square

If we run updates over the data only once, the theorem can be viewed as an online regret-bound. Since we assume that the training points are randomly drawn from an underlying distribution, by taking expectation with respect to the training data, the theorem essentially says that if we randomly stop the procedure after k-steps (uniform from 1 to m), then the performance of w_k is bounded by the best possible likelihood (with an arbitrary weight vector \bar{w}) plus two small penalty terms that

depend on the learning rate η . When $m \to \infty$, these penalty terms go to zero when we choose small η such that $\eta \to 0$ and $\eta m \to \infty$. If we go through the training data multiple times, with each training instance randomly drawn from the empirical distribution, then the method approximately solves Eq. 7. The theorem also implies that if we have a large amount of training data (m is large), then a small η does not slow down the convergence.

ACKNOWLEDGMENTS

This work was partially supported by DARPA and monitored by SPAWAR under contract No. N66001-99-2-8916. The paper has greatly profited from suggestions by the anonymous reviewers.

REFERENCES

- AL-ONAIZAN, Y., GE, N., LEE, Y.-S., PAPINENI, K., XIA, F., AND TILLMANN, C. 2004. IBM Site Report. In NIST 2004 MT Workshop. IBM, Alexandria, VA.
- Brown, P. F., Della Pietra, V. J., Della Pietra, S. A., and Mercer, R. L. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics* 19, 2, 263–311.
- Callison-Burch, C., Bannard, C., and Schroeder, J. Scaling Phrase-Based Statistical Machine Translation to Larger Corpora and Longer Phrases.
- Chiang, D. 2005. A Hierarchical Phrase-Based Model for Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics* (ACL'05). Association for Computational Linguistics, Ann Arbor, Michigan, 263–270.
- CHIANG, D., LOPEZ, A., MADNANI, N., MONZ, C., RESNIK, P., AND SUBOTIN, M. 2005. The Hiero Machine Translation System: Extensions, Evaluation, and Analysis. In Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Vancouver, British Columbia, Canada, 779–786.
- Collins, M. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proc. EMNLP'02*. Philadelphia,PA.
- DARROCH, J. AND RATCLIFF, D. 1972. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics* 43, 1470–1480.
- Della Pietra, S., Della Pietra, V., and Lafferty, J. 1997. Inducing Features of Random Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19, 4, 380–393.
- Deng, Y. and Byrne, W. 2005. HMM Word and Phrase Alignment for Statistical Machine Translation. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Vancouver, British Columbia, Canada, 169–176.
- KOEHN, P., OCH, F. J., AND MARCU, D. 2003. Statistical Phrase-Based Translation. In HLT-NAACL 2003: Main Proceedings. Association for Computational Linguistics, Edmonton, Alberta, Canada, 127–133.
- Kumar, S. and Byrne, W. 2003. A Weighted Finite State Transducer Implementation of the Alignment Template Model for Statistical Machine Translation. In *HLT-NAACL 2003: Main Proceedings*, M. Hearst and M. Ostendorf, Eds. Association for Computational Linguistics, Edmonton, Alberta, Canada, 142–149.
- Kumar, S. and Byrne, W. 2005. Local Phrase Reordering Models for Statistical Machine Translation. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Vancouver, British Columbia, Canada, 161–168.
- Kushner, H. J. and Yin, G. G. 1997. Stochastic approximation algorithms and applications. Springer-Verlag, New York.

- LAFFERTY, J., McCallum, A., and Pereira, F. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML-01*. 282–289.
- MARCU, D. 2001. Towards a Unified Approach to Memory- and Statistical-Based Machine Translation. In *Proc. of the 39th Annual Conf. of the Association for Computational Linguistics* (ACL 01). Toulouse, France.
- MARCU, D. AND WONG, D. 2002. A Phrase-Based, Joint Probability Model for Statistical Machine Translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Philadelphia, 133–139.
- McDonald, R., Crammer, K., and Pereira, F. 2005. Online Large-Margin Training of Dependency Parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. Association for Computational Linguistics, Ann Arbor, Michigan, 91–98.
- MOORE, R. C. 2005. A Discriminative Framework for Bilingual Word Alignment. In *Proceedings* of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Vancouver, British Columbia, Canada, 81–88.
- NAGATA, M., SAITO, K., YAMAMOTO, K., AND OHASHI, K. 2006. A Clustered Global Phrase Reordering Model for Statistical Machine Translation. In *Proceedings of the 21st International* Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, Sydney, Australia, 713– 720.
- Och, F. J. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings* of the 41st Annual Meeting of the Association for Computational Linguistics, E. Hinrichs and D. Roth, Eds. Association for Computational Linguistics, Sapporo, Japan, 160–167.
- Och, F.-J. and Ney, H. 2004. The Alignment Template Approach to Statistical Machine Translation. *Computational Linguistics* 30, 4, 417–450.
- Och, F.-J., Tillmann, C., and Ney, H. 1999. Improved Alignment Models for Statistical Machine Translation. In *Proc. of the Joint Conf. on Empirical Methods in Natural Language Processing* and Very Large Corpora (EMNLP/VLC 99). College Park, MD, 20–28.
- OCH ET AL. 2004. A Smorgasbord of Features for Statistical Machine Translation. In *Proceedings* of the Joint HLT and NAACL Conference (HLT 04). Boston, MA, 161–168.
- Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. 2002. BLEU: a Method for Automatic Evaluation of machine translation. In *Proc. of the 40th Annual Conf. of the Association for Computational Linguistics (ACL 02)*. Philadelphia, PA, 311–318.
- Schafer, C. and Yarowsky, D. 2003. Statistical Machine Translation Using Coercive Two-Level Syntactic Transduction. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, M. Collins and M. Steedman, Eds. Association for Computational Linguistics, Sapporo, Japan, 9–16.
- Shen, L., Sarkar, A., and Och, F.-J. 2004. Discriminative Reranking of Machine Translation. In *Proceedings of the Joint HLT and NAACL Conference (HLT 04)*. Boston, MA, 177–184.
- Taskar, B., Simon, L.-J., and Dan, K. 2005. A Discriminative Matching Approach to Word Alignment. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Vancouver, British Columbia, Canada, 73–80.
- Tillmann, C. 2003. A Projection Extension Algorithm for Statistical Machine Translation. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing (EMNLP 03)*. Sapporo, Japan, 1–8.
- TILLMANN, C. 2004. A Unigram Orientation Model for Statistical Machine Translation. In HLT-NAACL 2004: Short Papers, D. M. Susan Dumais and S. Roukos, Eds. Association for Computational Linguistics, Boston, Massachusetts, USA, 101–104.
- Tillmann, C. 2006. Efficient Dynamic Programming Search Algorithms for Phrase-based SMT. In Proceedings of the Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing (at HLT 06). New York City, NY, 9–16.
- TILLMANN, C. AND NEY, H. 2003. Word Re-ordering and a DP Beam Search Algorithm for Statistical Machine Translation. *Computational Linguistics* 29, 1, 97–133.

- TILLMANN, C. AND XIA, F. 2003. A Phrase-based Unigram Model for Statistical Machine Translation. In *Companion Vol. of HLT/NAACL 03*. Edmonton, Canada, 106–108.
- TILLMANN, C. AND ZHANG, T. 2005. A Localized Prediction Model for Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. Association for Computational Linguistics, Ann Arbor, Michigan, 557–564.
- TILLMANN, C. AND ZHANG, T. 2006. A Discriminative Global Training Algorithm for Statistical MT. In Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (ACL/COLING'05). Sydney, Australia, 721–728.
- UEFFING, N., OCH, F.-J., AND NEY, H. 2002. Generation of Word Graphs in Statistical Machine Translation. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing (EMNLP 02)*. Philadelphia, PA, 156–163.
- Venugopal, A., Vogel, S., and Waibel, A. 2003. Effective Phrase Translation Extraction from Alignment Models. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, E. Hinrichs and D. Roth, Eds. Association for Computational Linguistics, Sapporo, Japan, 319–326.
- Watanabe, T., Sumita, E., and Okuno, H. G. 2003. Chunk-Based Statistical Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, E. Hinrichs and D. Roth, Eds. Association of Computational Linguistics, Sapporo, Japan, 303–310.
- ZENS, R. AND NEY, H. 2004. Improvements in Phrase-Based Statistical Machine Translation. In HLT-NAACL 2004: Main Proceedings, D. M. Susan Dumais and S. Roukos, Eds. Association for Computational Linguistics, Boston, Massachusetts, USA, 257–264.
- ZHANG, T. 2004. Solving Large Scale Linear Prediction Problems Using Stochastic Gradient Descent Algorithms. In ICML 04, 919–926.
- ZHAO, B., GE, N., AND PAPINENI, K. 2005. Inner-Outer Bracket Models for Word Alignment using Hidden Blocks. In Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Vancouver, British Columbia, Canada, 177–184.

Received November 2005; Revised November 2006; accepted Month Year.