An End-to-End Discriminative Approach to Machine Translation

Percy Liang **Alexandre Bouchard-Côté** **Dan Klein** **Ben Taskar**
Computer Science Division, EECS Department
University of California at Berkeley
Berkeley, CA 94720
{pliang, bouchard, klein, taskar}@cs.berkeley.edu

Abstract

We present a perceptron-style discriminative approach to machine translation in which large feature sets can be exploited. Unlike discriminative reranking approaches, our system can take advantage of learned features in all stages of decoding. We first discuss several challenges to error-driven discriminative approaches. In particular, we explore different ways of updating parameters given a training example. We find that making frequent but smaller updates is preferable to making fewer but larger updates. Then, we discuss an array of features and show both how they quantitatively increase BLEU score and how they qualitatively interact on specific examples. One particular feature we investigate is a novel way to introduce learning into the initial phrase extraction process, which has previously been entirely heuristic.

1 Introduction

The generative, noisy-channel paradigm has historically served as the foundation for most of the work in statistical machine translation (Brown et al., 1994). At the same time, discriminative methods have provided substantial improvements over generative models on a wide range of NLP tasks. They allow one to easily encode domain knowledge in the form of features. Moreover, parameters are tuned to directly minimize error rather than to maximize joint likelihood, which may not correspond well to the task objective.

In this paper, we present an end-to-end discriminative approach to machine translation. The proposed system is phrase-based, as in Koehn et al. (2003), but uses an online perceptron training scheme to learn model parameters. Unlike minimum error rate training (Och, 2003), our system is able to exploit large numbers of specific features in the same manner as static reranking systems (Shen et al., 2004; Och et al., 2004). However, unlike static rerankers, our system does not rely on a baseline translation system. Instead, it updates based on its own n-best lists. As parameter estimates improve, the system produces better n-best lists, which can in turn enable better updates in future training iterations. In this paper, we focus on two aspects of the problem of discriminative translation: the inherent difficulty of learning from reference translations, and the challenge of engineering effective features for this task.

Discriminative learning from reference translations is inherently problematic because standard discriminative methods need to know which outputs are correct and which are not. However, a proposed translation that differs from a reference translation need not be incorrect. It may differ in word choice, literalness, or style, yet be fully acceptable. Pushing our system to avoid such alternate translations is undesirable. On the other hand, even if a system produces a reference translation, it may do so by abusing the hidden structure (sentence segmentation and alignment). We can therefore never be entirely sure whether or not a proposed output is safe to update towards. We discuss this issue in detail in Section 5, where we show that conservative updates (which push the system towards a local variant of the current prediction) are more effective than more aggressive updates (which try to directly update towards the reference).

The second major contribution of this work is an investigation of an array of features for our model. We show how our features quantitatively increase BLEU score, as well as how they qualitatively interact on specific examples. We first consider learning weights for individual phrases and part-of-speech patterns, showing gains from each. We then present a novel way to parameterize and introduce learning into the initial phrase extraction process. In particular, we introduce *alignment constellation* features, which allow us to weight phrases based on the word alignment pattern that led to their extraction. This kind of feature provides a potential way to initially extract phrases more aggressively and then later downweight undesirable patterns, essentially learning a weighted extraction heuristic. Finally, we use POS features to parameterize a distortion model in a limited distortion decoder (Zens and Ney, 2004; Tillmann and Zhang, 2005). We show that overall, BLEU score increases from 28.4 to 29.6 on French-English.

2 Approach

2.1 Translation as structured classification

Machine translation can be seen as a structured classification task, in which the goal is to learn a mapping from an input (French) sentence $\mathbf{x}$ to an output (English) sentence $\mathbf{y}$. Given this setup, discriminative methods allow us to define a broad class of features Φ that operate on $(\mathbf{x}, \mathbf{y})$. For example, some features would measure the fluency of $\mathbf{y}$ and others would measure the faithfulness of $\mathbf{y}$ as a translation of $\mathbf{x}$.

However, the translation task in this framework differs from traditional applications of discriminative structured classification such as POS tagging and parsing in a fundamental way. Whereas in POS tagging, there is a one-to-one correspondence between the words $\mathbf{x}$ and the tags $\mathbf{y}$, the correspondence between $\mathbf{x}$ and $\mathbf{y}$ in machine translation is not only much more complex, but is in fact unknown. Therefore, we introduce a hidden correspondence structure $\mathbf{h}$ and work with the feature vector $\Phi(\mathbf{x}, \mathbf{y}, \mathbf{h})$.

The phrase-based model of Koehn et al. (2003) is an instance of this framework. In their model, the correspondence $\mathbf{h}$ consists of (1) the segmentation of the input sentence into phrases, (2) the segmentation of the output sentence into the same number of phrases, and (3) a bijection between the input and output phrases. The feature vector $\Phi(\mathbf{x}, \mathbf{y}, \mathbf{h})$ contains four components: the log probability of the output sentence $\mathbf{y}$ under a language model, the score of translating $\mathbf{x}$ into $\mathbf{y}$ based on a phrase table, a distortion score, and a length penalty.[1] In Section 6, we vastly increase the number of features to take advantage of the full power of discriminative training.

Another example of this framework is the hierarchical model of Chiang (2005). In this model the correspondence $\mathbf{h}$ is a synchronous parse tree over input and output sentences, and features include the scores of various productions used in the tree.

Given features Φ and a corresponding set of parameters $\mathbf{w}$, a standard classification rule f is to return the highest scoring output sentence $\mathbf{y}$, maximizing over correspondences $\mathbf{h}$:

$$f(\mathbf{x}; \mathbf{w}) = \operatorname*{argmax}_{\mathbf{y},\mathbf{h}} \mathbf{w} \cdot \Phi(\mathbf{x}, \mathbf{y}, \mathbf{h}). \quad (1)$$

In the phrase-based model, computing the argmax exactly is intractable, so we approximate f with beam decoding.

2.2 Perceptron-based training

To tune the parameters $\mathbf{w}$ of the model, we use the averaged perceptron algorithm (Collins, 2002) because of its efficiency and past success on various NLP tasks (Collins and Roark, 2004; Roark et al., 2004). In principle, $\mathbf{w}$ could have been tuned by maximizing conditional probability or maximizing margin. However, these two options require either marginalization or numerical optimization, neither of which is tractable over the space of output sentences $\mathbf{y}$ and correspondences $\mathbf{h}$. In contrast, the perceptron algorithm requires only a decoder that computes $f(\mathbf{x}; \mathbf{w})$.

Recall the traditional perceptron update rule on an example $(\mathbf{x}_i, \mathbf{y}_i)$ is

$$\mathbf{w} \leftarrow \mathbf{w} + \Phi(\mathbf{x}_i, \mathbf{y}_\mathrm{t}) - \Phi(\mathbf{x}_i, \mathbf{y}_\mathrm{p}), \quad (2)$$

where $\mathbf{y}_\mathrm{t} = \mathbf{y}_i$ is the *target output* and $\mathbf{y}_\mathrm{p} = f(\mathbf{x}_i; \mathbf{w}) = \mathrm{argmax}_{\mathbf{y}}\, \mathbf{w} \cdot \Phi(\mathbf{x}_i, \mathbf{y})$ is the prediction using the current parameters $\mathbf{w}$.

We adapt this update rule to work with hidden variables as follows:

$$\mathbf{w} \leftarrow \mathbf{w} + \Phi(\mathbf{x}_i, \mathbf{y}_\mathrm{t}, \mathbf{h}_\mathrm{t}) - \Phi(\mathbf{x}_i, \mathbf{y}_\mathrm{p}, \mathbf{h}_\mathrm{p}), \quad (3)$$

where $(\mathbf{y}_\mathrm{p}, \mathbf{h}_\mathrm{p})$ is the argmax computation in Equation 1, and $(\mathbf{y}_\mathrm{t}, \mathbf{h}_\mathrm{t})$ is the *target* that we update towards. If $(\mathbf{y}_\mathrm{t}, \mathbf{h}_\mathrm{t})$ is the same argmax computation with the additional constraint that $\mathbf{y}_\mathrm{t} = \mathbf{y}_i$, then Equation 3 can be interpreted as a Viterbi approximation to the stochastic gradient

$$\mathbb{E}_{P(\mathbf{h} \mid \mathbf{x}_i, \mathbf{y}_i; \mathbf{w})} \Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}) - \mathbb{E}_{P(\mathbf{y}, \mathbf{h} \mid \mathbf{x}_i; \mathbf{w})} \Phi(\mathbf{x}_i, \mathbf{y}, \mathbf{h})$$

for the following conditional likelihood objective:

$$P(\mathbf{y}_i \mid \mathbf{x}_i) \propto \sum_{\mathbf{h}} \exp(\mathbf{w} \cdot \Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h})).$$

[1] More components can be added to the feature vector if additional language models or phrase tables are available.

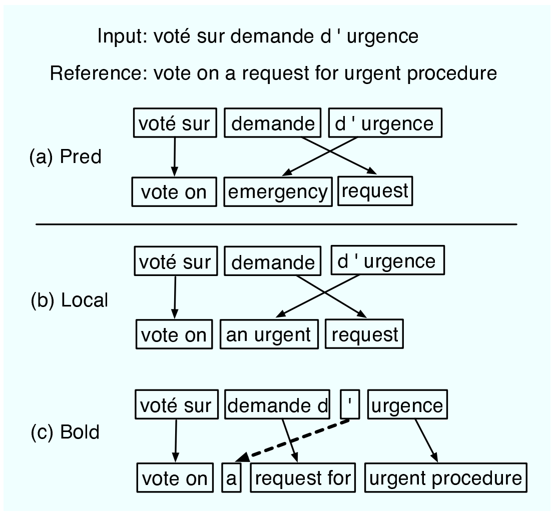


Figure 1: Given the current prediction (a), there are two possible updates, local (b) and bold (c). Although the bold update (c) reaches the reference translation, a bad correspondence is used. The local update (b) does not reach the reference, but is more reasonable than (c).

Discriminative training with hidden variables has been handled in this probabilistic framework (Quattoni et al., 2004; Koo and Collins, 2005), but we choose Equation 3 for efficiency.

It turns out that using the Viterbi approximation (which we call *bold* updating) is not always the best strategy. To appreciate the difficulty, consider the example in Figure 1. Suppose we make the prediction (a) with the current set of parameters. There are often several acceptable output translations $\mathbf{y}$, for example, (b) and (c). Since (c)'s output matches the reference translation, should we update towards (c)? In this case, the answer is negative. The problem with (c) is that the correspondence $\mathbf{h}$ contains an incorrect alignment (*'*, *a*). However, since $\mathbf{h}$ is unobserved, the training procedure has no way of knowing this. While the output in (b) is farther from the reference, its correspondence $\mathbf{h}$ is much more reasonable. In short, it does not suffice for $\mathbf{y}_t$ to be good; both $\mathbf{y}_t$ and $\mathbf{h}_t$ need to be good. A major challenge in using the perceptron algorithm for machine translation is determining the target $(\mathbf{y}_t, \mathbf{h}_t)$ in Equation 3. Section 5 discusses possible targets to update towards.

3 Dataset

Our experiments were done on the French-English portion of the Europarl corpus (Koehn, 2002), which consists of European parliamentary proceedings from 1996 to 2003.

Dataset	TRAIN	DEV	TEST
Years	’99–’01	’02	’03
# sentences	67K	first 1K	first 1K
# words (unk.)	715K	10.4K (35)	10.8K (48)

Table 1: The Europarl dataset split we used and various statistics on length 5–15 sentences. The number of French word tokens is given, along with the number that were not seen among the 414K total sentences in TRAIN (which includes all lengths).

We split the data into three sets according to Table 1. TRAIN served two purposes: it was used to construct the features, and the 5–15 length sentences were used for tuning the parameters of those features. DEV, which consisted of the first 1K length 5–15 sentences in 2002, was used to evaluate the performance of the system as we developed it. Note that the DEV set was not used to tune any parameters; tuning was done exclusively on TRAIN. At the end we ran our models once on TEST to get final numbers.[2]

4 Models

Our experiments used phrase-based models (Koehn et al., 2003), which require a translation table and language model for decoding and feature computation. To facilitate comparison with previous work, we created the translation tables using the same techniques as Koehn et al. (2003).[3] The language model was a Kneser-Ney interpolated trigram model generated using the SRILM toolkit (Stolcke, 2002). We built our own phrase-based beam decoder that can handle arbitrary features.[4] The contributions of features are incrementally added into the score as decoding

[2]We also experimented with several combinations of jackknifing to prevent overfitting, in which we selected features on TRAIN-OLD (1996–1998 Europarl corpus) and tuned the parameters on TRAIN, or vice-versa. However, it turned out that using TRAIN-OLD was suboptimal since that data is less relevant to DEV. Another alternative is to combine TRAIN-OLD and TRAIN into one dual-purpose dataset. The differences between this and our current approach were inconclusive.

[3]In other words, we used GIZA++ to construct a word alignment in each direction and a growth heuristic to combine them. We extracted all the substrings that are closed under this high-quality word alignment and computed surface statistics from cooccurrences counts.

[4]In our experiments, we used a beam size of 10, which we found to be only slightly worse than using a beam of 100.

proceeds.

We experimented with two levels of distortion: *monotonic*, where the phrasal alignment is monotonic (but word reordering is still possible within a phrase) and *limited distortion*, where only adjacent phrases are allowed to exchange positions (Zens and Ney, 2004). In the future, we plan to explore our discriminative framework on a full distortion model (Koehn et al., 2003) or even a hierarchical model (Chiang, 2005).

Throughout the following experiments, we trained the perceptron algorithm for 10 iterations. The weights were initialized to 1 on the translation table, 1 on the language model (the blanket features in Section 6), and 0 elsewhere. The next two sections give experiments on the two key components of a discriminative machine translation system: choosing the proper update strategy (Section 5) and including powerful features (Section 6).

5 Update strategies

This section describes the importance of choosing a good update strategy—the difference in BLEU score can be as large as 1.2 between different strategies. An update strategy specifies the *target* $(\mathbf{y}_\mathrm{t}, \mathbf{h}_\mathrm{t})$ that we update towards (Equation 3) given the current set of parameters and a provided reference translation $(\mathbf{x}_i, \mathbf{y}_i)$. As mentioned in Section 2.2, faithful output (i.e. $\mathbf{y}_\mathrm{t} = \mathbf{y}_i$) does not imply that updating towards $(\mathbf{y}_\mathrm{t}, \mathbf{h}_\mathrm{t})$ is desirable. In fact, such a constrained target might not even be *reachable* by the decoder, for example, if the reference is very non-literal.

We explored the following three ways to choose the target $(\mathbf{y}_\mathrm{t}, \mathbf{h}_\mathrm{t})$:

- *Bold updating*: Update towards the highest scoring option $(\mathbf{y}, \mathbf{h})$, where $\mathbf{y}$ is constrained to be the reference $\mathbf{y}_i$ but $\mathbf{h}$ is unconstrained. Examples not reachable by the decoder are skipped.

- *Local updating*: Generate an n-best list using the current parameters. Update towards the option with the highest BLEU score.[5]

[5]Since BLEU score (k-BLEU with $k = 4$) involves computing a geometric mean over i-grams, $i = 1, \ldots, k$, it is zero if the translation does not have at least one k-gram in common with the reference translation. Since a BLEU score of zero is both unhelpful for choosing from the n-best and common when computed on just a single example, we instead used a smoothed version for choosing the target: $\sum_{i=1}^{4} \frac{i\text{-BLEU}(x,y)}{2^{4-i+1}}$. We still report NIST's usual 4-gram BLEU.

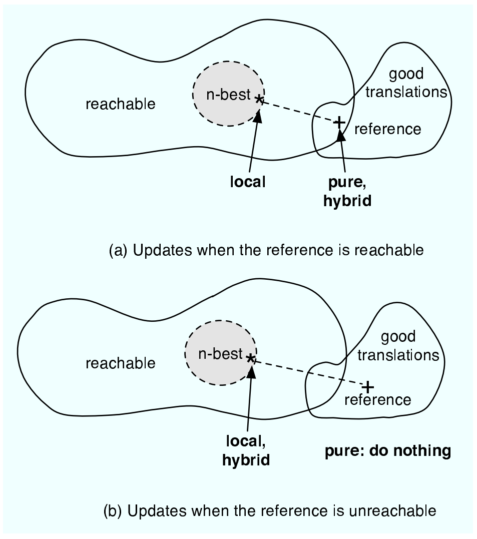


Figure 2: The three update strategies under two scenarios.

- *Hybrid updating*: Do a bold update if the reference is reachable. Otherwise, do a local update.

Figure 2 shows the space of translations schematically. On each training example, our decoder produces an n-best list. The reference translation may or may not be reachable.

Bold updating most resembles the traditional perceptron update rule (Equation 2). We are ensured that the target output $\mathbf{y}$ will be correct, although the correspondence $\mathbf{h}$ might be bad. Another weakness of bold updating is that we might not make full use of the training data.

Local updating uses every example, but its steps are more cautious. It can be viewed as "dynamic reranking," where parameters are updated using the best option on the n-best list, similar to standard static reranking. The key difference is that, unlike static reranking, the parameter updates propagate back to the baseline classifier, so that the n-best list improves over time. In this regard, dynamic reranking remedies one of the main weaknesses of static reranking, which is that the performance of the system is directly limited by the quality of the baseline classifier.

Hybrid updating combines the two strategies: it makes full use of the training data as in local updating, but still tries to make swift progress towards the reference translation as in bold updating.

We conducted experiments to see which of the updating strategies worked best. We trained on

Decoder	Bold	Local	Hybrid
Monotonic	34.3	**34.6**	34.5
Limited distortion	33.5	**34.7**	33.6

Table 2: Comparison of BLEU scores between different updating strategies for the monotonic and limited distortion decoders on DEV.

5000 of the 67K available examples, using the BLANKET+LEX+POS feature set (Section 6). Table 2 shows that local updating is the most effective, especially when using the limited distortion decoder.

In bold updating, only a small fraction of the 5000 examples (1296 for the monotonic decoder and 1601 for the limited distortion decoder) had reachable reference translations, and, therefore, contributed to parameter updates. One might therefore hypothesize that local updating performs better simply because it is able to leverage more data. This is not the full story, however, since the hybrid approach (which makes the same number of updates) performs significantly worse than local updating when using the limited distortion decoder.

To see the problem with bold updating, recall the example in Figure 1. Bold updating tries to reach the reference at all costs, even if it means abusing the hidden correspondence in the process. In the example, the alignment (’, *a*) is unreasonable, but the algorithm has no way to recognize this. Local updating is much more stable since it only updates towards sentences in the n-best list.

When using the limited distortion decoder, bold updating is even more problematic because the added flexibility of phrase swaps allows more preposterous alignments to be produced. Limited distortion decoding actually performs *worse* than monotonic decoding with bold updating, but better with local updating.

Another difference between bold updating and local updating is that the BLEU score on the training data is dramatically higher for bold updating than for local (or hybrid) updating: 80 for the former versus 40 for the latter. This is not surprising given that bold updating aggressively tries to obtain the references. However, what is surprising is that although bold updating appears to be overfitting severely, its BLEU score on the DEV does not suffer much in the monotonic case.

Model	DEV BLEU	TEST BLEU
Monotonic		
BLANKET (untuned)	33.0	28.3
BLANKET	33.4	28.4
BLANKET+LEX	35.0	29.2
BLANKET+LEX+POS	**35.3**	**29.6**
Pharaoh (MERT)	34.5	28.8
Full-distortion		
Pharaoh (MERT)	34.9	29.5

Table 3: Main results on our system with different feature sets compared to minimum error-rate trained Pharaoh.

6 Features

This section shows that by adding an array of expressive features and discriminatively learning their weights, we can obtain a 2.3 increase in BLEU score on DEV. We add these features incrementally, first tuning blanket features (Section 6.1), then adding lexical features (Section 6.2), and finally adding part-of-speech (POS) features (Section 6.3). Table 3 summarizes the performance gains.

For the experiments in this section, we used the local updating strategy and the monotonic decoder for efficiency. We train on all 67K of the length 5–15 sentences in TRAIN.[6]

6.1 Blanket features

The blanket features (BLANKET) consist of the translation log-probability and the language model log-probability, which are two of the components of the Pharaoh model (Section 2.1). After discriminative training, the relative weight of these two features is roughly 2:1, resulting in a BLEU score increase from 33.0 (setting both weights to 1) to 33.4.

The following simple example gives a flavor of the discriminative approach. The untuned system translated the French phrase *trente-cinq langues* into *five languages* in a DEV example. Although the probability $P(\textit{five} \mid \textit{trente-cinq}) = 0.065$ is rightly much smaller than $P(\textit{thirty-five} \mid \textit{trente-cinq}) = 0.279$, the language model favors *five languages* over *thirty-five languages*. The trained system downweights the language model and recovers the correct translation.

[6]We used sentences of length 5–15 to facilitate comparisons with Koehn et al. (2003) and to enable rapid experimentation with various feature sets. Experiments on sentences of length 5–50 showed similar gains in performance.

6.2 Lexical features

The blanket features provide a rough guide for translation, but they are far too coarse to fix specific mistakes. We therefore add *lexical features* (LEX) to allow for more fine-grained control. These features come in two varieties. Lexical phrase features indicate the presence of a specific translation phrase, such as (*y a-t-il*, *are there*), and lexical language model features indicate the presence of a specific output n-gram, such as *of the*. Lexical language model features have been exploited successfully in discriminative language modeling to improve speech recognition performance (Roark et al., 2004). We confirm the utility of the two kinds of lexical features: BLANKET+LEX achieves a BLEU score of 35.0, an improvement of 1.6 over BLANKET.

To understand the effect of adding lexical features, consider the ten with highest and lowest weights after training:

64	any comments ?	-55	(des, of)
63	(y a-t-il, are there)	-52	(y a-t-il, are there any)
62	there any comments	-42	there any of
57	any comments	-39	of comments
46	(des, any)	-38	of comments ?

These features can in fact be traced back to the following example:

Input	y a-t-il	des	observations ?
B	**are there any**	**of**	comments ?
B+L	**are there**	**any**	comments ?

The second and third rows are the outputs of BLANKET (wrong) and BLANKET+LEX (correct), respectively. The correction can be accredited to two changes in feature weights. First, the lexical feature (*y a-t-il*, *are there any*) has been assigned a negative weight and (*y a-t-il*, *are there*) a positive weight to counter the fact that the former phrase incorrectly had a higher score in the original translation table. Second, (*des*, *of*) is preferred over (*des*, *any*), even though the former is a better translation in isolation. This apparent degradation causes no problems, because when *des* should actually be translated to *of*, these words are usually embedded in larger phrases, in which case the isolated translation probability plays no role.

Another example of a related phenomenon is the following:

Input	...	pour cela que	j ' ai	voté favorablement	.
B	...	for that	**i have**	voted in favour	.
B+L	...	for this reason	**i**	voted in favour	.

Counterintuitively, the phrase pair (*j ' ai*, *I have*) ends up with a very negative weight. The reason behind this is that in French, *j ' ai* is often used in a paraphrastic construction which should be translated into the simple past in English. For that to happen, *j ' ai* needs to be aligned with *I*. Since (*j ' ai*, *I*) has a small score compare to (*j ' ai*, *I have*) in the original translation table, downweighting the latter pair allows this sentence to be translated correctly.

A general trend is that literal phrase translations are downweighted. Lessening the pressure to literally translate certain phrases allows the language model to fill in the gaps appropriately with suitable non-literal translations. This point highlights the strength of discriminative training: weights are jointly tuned to account for the intricate interactions between overlapping phrases, which is something not achievable by estimating the weights directly from surface statistics.

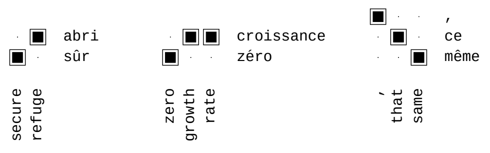
6.3 Part-of-speech features

While lexical features are useful for eliminating specific errors, they have limited ability to generalize to related phrases. This suggests the use of similar features which are abstracted to the POS level.[7] In our experiments, we used the TreeTagger POS tagger (Schmid, 1994), which ships pretrained on several languages, to map each word to its majority POS tag. We could also relatively easily base our features on context-dependent POS tags: the entire input sentence is available before decoding begins, and the output sentence is decoded left-to-right and could be tagged incrementally.

Where we had lexical phrase features, such as (*la réalisation du droit*, *the right*), we now also have their POS abstractions, for instance (DT NN IN NN, DT NN). This phrase pair is undesirable, not because of particular lexical facts about *la réalisation*, but because dropping a nominal head is generally to be avoided. The lexical language model features have similar POS counterparts. With these two kinds of POS features, we obtained an 0.3 increase in BLEU score from BLANKET+LEX to BLANKET+LEX+POS.

Finally, when we use the limited distortion decoder, it is important to learn when to swap adjacent phrases. Unlike Pharaoh, which simply has a uniform penalty for swaps, we would like to use context—in particular, POS information. For example, we would like to know that if a (JJ, JJ)

[7]We also tried using word clusters (Brown et al., 1992) instead of POS but found that POS was more helpful.



(a) (b) (c)

Figure 3: Three constellation features with example phrase pairs. Constellations (a) and (b) have large positive weights and (c) has a large negative weight.

phrase is constructed after a (NN, NN) phrase, they are reasonable candidates for swapping because of regular word-order differences between French and English. While the bulk of our results are presented for the monotonic case, the limited distortion results of Table 2 use these lexical swap features; without parameterized swap features, accuracy was below the untuned monotonic baseline.

An interesting statistic is the number of nonzero feature weights that were learned using each feature set. BLANKET has only 4 features, while BLANKET+LEX has 1.55 million features.[8] Remarkably, BLANKET+LEX+POS has fewer features—only 1.24 million. This is an effect of generalization ability—POS information somewhat reduces the need for specific lexical features.

6.4 Alignment constellation features

Koehn et al. (2003) demonstrated that choosing the appropriate heuristic for extracting phrases is very important. They showed that the difference in BLEU score between various heuristics was as large as 2.0.

The process of phrase extraction is difficult to optimize in a non-discriminative setting: many heuristics have been proposed (Koehn et al., 2003), but it is not obvious which one should be chosen for a given language pair. We propose a natural way to handle this part of the translation pipeline. The idea is to push the learning process all the way down to the phrase extraction by parameterizing the phrase extraction heuristic itself.

The heuristics in Koehn et al. (2003) decide whether to extract a given phrase pair based on the underlying word alignments (see Figure 3 for three examples), which we call *constellations*. Since we do not know which constellations correspond to good phrase pairs, we introduce an alignment constellation feature to indicate the presence of a particular alignment constellation.[9]

Features	-CONST	+CONST
BLANKET	31.8	32.2
BLANKET+LEX	32.2	32.5
BLANKET+LEX+POS	32.3	32.5

Table 4: DEV BLEU score increase resulting from adding constellation features.

Table 4 details the effect of adding constellation features on top of our previous feature sets.[10] We get a minor increase in BLEU score from each feature set, although there is no gain by adding POS features in addition to constellation features, probably because POS and constellation features provide redundant information for French-English translations.

It is interesting to look at the constellations with highest and lowest weights, which are perhaps surprising at first glance. At the top of the list are word inversions (Figure 3 (a) and (b)), while long monotonic constellations fall at the bottom of the list (c). Although monotonic translations are much more frequent than word inversions in our dataset, when translations are monotonic, shorter segmentations are preferred. This phenomenon is another manifestation of the complex interaction of phrase segmentations.

7 Final results

The last column of Table 3 shows the performance of our methods on the final TEST set. Our best test BLEU score is 29.6 using BLANKET+LEX+POS, an increase of 1.3 BLEU over our untuned feature set BLANKET. The discrepancy between DEV performance and TEST performance is due to temporal distance from TRAIN and high variance in BLEU score.[11]

We also compared our model with Pharaoh (Koehn et al., 2003). We tuned Pharaoh's four parameters using minimum error rate training (Och, 2003) on DEV.[12] We obtained an increase of 0.8

[8]Both the language model and translation table components have two features, one for known words and one for unknown words.

[9]As in the POS features, we map each phrase pair to its majority constellation.

[10]Due to time constraints, we ran these experiments on 5000 training examples using bold updating.

[11]For example, the DEV BLEU score for BLANKET+LEX ranges from 28.6 to 33.2, depending on which block of 1000 sentences we chose.

[12]We used the training scripts from the 2006 MT Shared Task. We still tuned our model parameters on TRAIN and

BLEU over the Pharaoh, run with the monotone flag.[13] Even though we are using a monotonic decoder, our best results are still slightly better than the version of Pharaoh that permits arbitrary distortion.

8 Related work

In machine translation, most discriminative approaches currently fall into two general categories. The first approach is to reuse the components of a generative model, but tune their relative weights in a discriminative fashion (Och and Ney, 2002; Och, 2003; Chiang, 2005). This approach only works in practice with a small handful of parameters.

The second approach is to use reranking, in which a baseline classifier generates an n-best list of candidate translations, and a separate discriminative classifier chooses amongst them (Shen et al., 2004; Och et al., 2004). The major limitation of a reranking system is its dependence on the underlying baseline system, which bounds the potential improvement from discriminative training. In machine translation, this limitation is a real concern; it is common for all translations on moderately-sized n-best lists to be of poor quality. For instance, Och et al. (2004) reported that a 1000-best list was required to achieve performance gains from reranking. In contrast, the decoder in our system can use the feature weights learned in the previous iteration.

Tillmann and Zhang (2005) present a discriminative approach based on local models. Their formulation explicitly decomposed the score of a translation into a sequence of local decisions, while our formulation allows global estimation.

9 Conclusion

We have presented a novel end-to-end discriminative system for machine translation. We studied update strategies, an important issue in online discriminative training for MT, and conclude that making many smaller (conservative) updates is better than making few large (aggressive) updates. We also investigated the effect of adding many expressive features, which yielded a 0.8 increase in BLEU score over monotonic Pharaoh.

Acknowledgments We would like to thank our reviewers for their comments. This work was supported by a FQRNT fellowship to second author and a Microsoft Research New Faculty Fellowship to the third author.

only used DEV to optimize the number of training iterations.

[13]This result is significant with p-value 0.0585 based on approximate randomization (Riezler and Maxwell, 2005).

References

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jennifer C. Lai, and Robert L. Mercer. 1992. Class-Based n-gram Models of Natural Language. *Computational Linguistics*, 18(4):467–479.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1994. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19:263–311.

David Chiang. 2005. A Hierarchical Phrase-Based Model for Statistical Machine Translation. In *ACL 2005*.

Michael Collins and Brian Roark. 2004. Incremental Parsing with the Perceptron Algorithm. In *ACL 2004*.

Michael Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *EMNLP 2002*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *HLT-NAACL 2003*.

Philipp Koehn. 2002. Europarl: A Multilingual Corpus for Evaluation of Machine Translation.

Terry Koo and Michael Collins. 2005. Hidden-Variable Models for Discriminative Reranking. In *EMNLP 2005*.

Franz Josef Och and Hermann Ney. 2002. Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. In *ACL 2002*.

Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, and Anoop Sarkar. 2004. A Smorgasbord of Features for Statistical Machine Translation. In *HLT-NAACL 2004*.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *ACL 2003*.

Ariadna Quattoni, Michael Collins, and Trevor Darrell. 2004. Conditional Random Fields for Object Recognition. In *NIPS 2004*.

Stefan Riezler and John T. Maxwell. 2005. On Some Pitfalls in Automatic Evaluation and Significance Testing for MT. In *Workshop on Intrinsic and Extrinsic Evaluation Methods for MT and Summarization (MTSE)*.

Brian Roark, Murat Saraclar, Michael Collins, and Mark Johnson. 2004. Discriminative Language Modeling with Conditional Random Fields and the Perceptron Algorithm. In *ACL 2004*.

Helmut Schmid. 1994. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *International Conference on New Methods in Language Processing*.

Libin Shen, Anoop Sarkar, and Franz Josef Och. 2004. Discriminative Reranking for Machine Translation. In *HLT-NAACL 2004*.

Andreas Stolcke. 2002. SRILM An Extensible Language Modeling Toolkit. In *ICSLP 2002*.

Christoph Tillmann and Tong Zhang. 2005. A Localized Prediction Model for Statistical Machine Translation. In *ACL 2005*.

Richard Zens and Hermann Ney. 2004. Improvements in Statistical Phrase-Based Translation. In *HLT-NAACL 2004*.