A Sequence Alignment Model Based on the Averaged Perceptron

Dayne Freitag
Fair Isaac Corporation
3661 Valley Centre Drive
San Diego, CA 92130, USA
DayneFreitag@fairisaac.com

Shahram Khadivi
Lehrstuhl für Informatik 6
Computer Science Department
RWTH Aachen University
D-52056 Aachen, Germany
khadivi@cs.rwth-aachen.de

Abstract

We describe a discriminatively trained sequence alignment model based on the averaged perceptron. In common with other approaches to sequence modeling using perceptrons, and in contrast with comparable generative models, this model permits and transparently exploits arbitrary features of input strings. The simplicity of perceptron training lends more versatility than comparable approaches, allowing the model to be applied to a variety of problem types for which a learned edit model might be useful. We enumerate some of these problem types, describe a training procedure for each, and evaluate the model's performance on several problems. We show that the proposed model performs at least as well as an approach based on statistical machine translation on two problems of name transliteration, and provide evidence that the combination of the two approaches promises further improvement.

1 Introduction

Sequence alignment is a problem that crops up in many forms, both in computational linguistics (CL) and in other endeavors. The ability to find an optimal alignment between two sequences has found application in a number of areas of CL, including phonetic modeling (Ristad and Yianilos, 1998), name transcription (Huang et al., 2004), and duplicate detection or information integration (Bilenko and Mooney, 2003; McCallum et al., 2005). Sequence alignment is a member of a broader class of problems which we might call *sequence transduction*, to which one of the core CL challenges, machine translation, belongs.

Under the assumption that one string (the *target*) is produced through a series of local edits to another string (the *source*), and given an edit cost matrix, the optimal sequence of edits can be efficiently computed through dynamic programming (Needleman and Wunsch, 1970). While the cost matrix traditionally has been set by hand, several recent papers have proposed determining edit costs empirically. These proposals arise from a variety of learning paradigms, including generative models (Ristad and Yianilos, 1998; Bilenko and Mooney, 2003), conditional random fields (McCallum et al., 2005), maximum-margin methods (Joachims, 2003), and gradient boosting (Parker et al., 2006). While approaches based on generative models support only limited feature engineering, discriminative approaches share the advantage of allowing arbitrary features of the input sequences.

We describe a new sequence alignment model based on the averaged perceptron (Collins, 2002), which shares with the above approaches the ability to exploit arbitrary features of the input sequences, but is distinguished from them by its relative simplicity and the incremental character of its training procedure. The fact that it is an online algorithm makes it straightforward to adapt to a range of problems. To show this, we evaluate the approach on several different tasks, some of them merely illustrative, but some with clear practical significance,

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 238–247, Prague, June 2007.

particularly the problem of named entity transcription.

2 The Algorithm

2.1 The Formalism

Suppose we are given two sequences, $\boldsymbol{s}_1^m \in \Sigma_s^*$ and $\boldsymbol{t}_1^n \in \Sigma_t^*$. We desire a real-valued function $A(\boldsymbol{s}, \boldsymbol{t})$ which assigns high scores to pairs $\boldsymbol{s}, \boldsymbol{t}$ with high *affinity*, where affinity is an application-specific notion (e.g., $\boldsymbol{t}$ is a likely phoneme sequence represented by the letter sequence $\boldsymbol{s}$). If we stipulate that this score is the sum of the individual scores of a series of edits, we can find the highest-scoring such series through a generalization of the standard edit distance:

$$A(\boldsymbol{s}_1^i, \boldsymbol{t}_1^j) = \max \left\{ \begin{array}{l} a_{\epsilon,t_j}(\boldsymbol{s}, i, \boldsymbol{t}, j) + A(\boldsymbol{s}_1^i, \boldsymbol{t}_1^{j-1}) \\ a_{s_i,\epsilon}(\boldsymbol{s}, i, \boldsymbol{t}, j) + A(\boldsymbol{s}_1^{i-1}, \boldsymbol{t}_1^j) \\ a_{s_i,t_j}(\boldsymbol{s}, i, \boldsymbol{t}, j) + A(\boldsymbol{s}_1^{i-1}, \boldsymbol{t}_1^{j-1}) \end{array} \right. \quad (1)$$

with $A(\emptyset, \emptyset) = 0$. The function $a_{s_i,t_j}(\boldsymbol{s}, i, \boldsymbol{t}, j)$ represents the score of substituting t_j for s_i; a_{ϵ,t_j} and $a_{s_i,\epsilon}$ represent insertion and deletion, respectively. If we assume constant-time computation of primitive edit costs, this recursive definition of A allows us to find the highest scoring series of edits for a given sequence pair in time proportional to the product of their lengths. Note that a is indexed by the characters involved in an edit (i.e., inserting 'e' generally has a different cost than inserting 's'). Note further that the score associated with a particular operation may depend on any features computable from the respective positions in the two sequences.

In the experiments reported in this paper, we assume that each local function a is defined in terms of $p + q$ features, $\{f_1, \cdots, f_p, f_{p+1}, \cdots, f_{p+q}\}$, and that these features have the functional form $\Sigma^* \times \mathcal{N} \mapsto \mathcal{R}$. In other words, each feature takes a sequence and an index and returns a real value. The first p features are defined over sequences from the source alphabet, while the remaining q are defined over the target alphabet.[1] In this paper we use character n-gram indicator features.

[1] Of course, features that depend jointly on both sequences may also be of interest.

Given a set $\boldsymbol{S}$ of source sequences
$V \leftarrow []$, an empty list
$\alpha \leftarrow \boldsymbol{0}$, a weight vector
for some number of iterations **do**
    for $\boldsymbol{s}$ in $\boldsymbol{S}$ **do**
        Pick $\boldsymbol{t}, \boldsymbol{t'}$, $\boldsymbol{t}$ having higher affinity with $\boldsymbol{s}$
        $\langle \boldsymbol{e}, v \rangle \leftarrow A_\alpha(\boldsymbol{s}, \boldsymbol{t})$
        $\langle \boldsymbol{e'}, v' \rangle \leftarrow A_\alpha(\boldsymbol{s}, \boldsymbol{t'})$
        if $v' \geq v$ **then**
            $\alpha \leftarrow \alpha + \Phi(\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{e}) - \Phi(\boldsymbol{s}, \boldsymbol{t'}, \boldsymbol{e'})$
        end if
        Append α to V
    end for
end for
Return the mean α from V

Table 1: The training algorithm. A_α is the affinity function under model parameters α, returning edit sequence $\boldsymbol{e}$ and score v.

The score of a particular edit is a linear combination of the corresponding feature values:

$$a(\boldsymbol{s}, i, \boldsymbol{t}, j) = \sum_{k=1}^{p} \alpha_k \cdot f_k(\boldsymbol{s}, i) + \sum_{k=p+1}^{p+q} \alpha_k \cdot f_k(\boldsymbol{t}, j) \quad (2)$$

The weights α_k are what we seek to optimize in order to tune the model for our particular application.

2.2 A Perceptron-Based Edit Model

In this section we present a general-purpose extension of perceptron training for sequence labeling, due to Collins (2002), to the problem of sequence alignment. Take α to be a model parameterization, and let $A_\alpha(\boldsymbol{s}, \boldsymbol{t})$ return an optimal edit sequence $\boldsymbol{e}$, with its score v, given input sequences $\boldsymbol{s}$ and $\boldsymbol{t}$ under α. Elements of sequence $\boldsymbol{e}$ are character pairs $\langle c_s, c_t \rangle$, with $c_s \in \Sigma_s \cup \{\epsilon\}$ and $c_t \in \Sigma_t \cup \{\epsilon\}$, where ϵ represents the empty string. Let $\Phi(\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{e})$ be a feature vector, having the same dimensionality as α, for a source, target, and corresponding edit sequence. This feature vector is the sum of feature vectors at each point in $\boldsymbol{e}$ as it is played out along input sequences $\boldsymbol{s}$ and $\boldsymbol{t}$.

Table 1 shows the basic algorithm. Starting with a zero parameter vector, we iterate through the collection of source sequences. For each sequence, we pick two target sequences having unequal affinity with the source sequence (Line 6). If the scores returned by our current model (Lines 7 and 8) agree with our ordering, we do nothing. Otherwise, we update the model using the perceptron training rule (Line 10). Ultimately, we return $\boldsymbol{\alpha}$ averaged over all datapoint presentations.

2.3 Training Modes

The algorithm presented in Table 1 does not specify how the two target sequences $\boldsymbol{t}$ and $\boldsymbol{t'}$ are to be chosen in Line 6. The answer to this question depends on the application. There are fundamentally two settings, depending on whether or not target strings are drawn from the same set as source strings; we will call the setting in which source and target strings inhabit the same set the *affinity* setting, and refer to the the case where they form different sets as the *transduction* setting. Here, we sketch four problem scenarios, two from each setting, and specify a target selection procedure appropriate for each.

Affinity, ranking. The task poses a latent affinity between strings, but we can measure it only indirectly. In particular, we can order *some* of the target sequences according to their affinity with a source sequence $\boldsymbol{s}$. In this case, we train as follows: Order a sample of the target sequences according to this partial order. Let $\boldsymbol{t}$ and $\boldsymbol{t'}$ be two sequences from this order, such that $\boldsymbol{t}$ is ordered higher than $\boldsymbol{t'}$.

Affinity, classification. The sequences in Σ^* can be grouped into classes, and we wish the model to assign high affinity to co-members of a class and low affinity to members of different classes. Train as follows: For each $\boldsymbol{s}$, sample $\boldsymbol{t}$ from among its co-members and $\boldsymbol{t'}$ from among the members of other classes.

Transduction, ranking. The data is presented as source-target pairs, where each $\boldsymbol{t}$ is a transduction of the corresponding $\boldsymbol{s}$. We wish to learn a model which, given a novel $\boldsymbol{s}$, will enable us to rank candidate transductions. Train as follows: Given $\boldsymbol{s}$, let $\boldsymbol{t}$ be the target sequence provided to us. Sample $\boldsymbol{t'}$ from among the other target sequences.

Transduction, generation. We are again given source-target pairs. We wish to learn to generate a probable target string, given a novel source string. Train as follows: *Generate* a $\boldsymbol{t'}$ that is approximately optimal according to the current model. Note that since edit decisions are based in part on (arbitrary) features of the target sequence, and since generation involves construction of the target sequence, it is not uncommon for a greedy generator to make edit decisions which are locally optimal, but which result several edits later in a partially constructed sequence in which no good edits are available. Thus, the problem of generation does not correspond to a simple recurrence relation like Equation 1. Consequently, we experimented with several heuristic approaches to generation and found that a beam search works well.

`edit‸ing`	$\rightarrow$	`STR‸INGS`
$f_{s,it\_}$		$f_{t,TR\_}$
$f_{s,t\_}$		$f_{t,R\_}$
$f_{s,\_in}$		$f_{\emptyset}$
$f_{s,\_i}$		

Table 2: Features with non-zero value for an example string pair and a model of order 2.

3 Evaluation

To establish the effectiveness of the model, we trained it on a range of problems, including instances of each of the four settings enumerated above. Problems ranged from the merely illustrative to a non-trivial application of computational linguistics.

3.1 Feature Construction

Without exception, the features we provide to the algorithm are the same in all experiments. Given a user-specified *order* k, we define a Boolean feature for every distinct character gram observed in the data of length k or smaller. Recall that there are two disjoint sets of features, those defined over strings drawn from the source and target alphabets, respectively. Given a source string and index, those features have value 1 whose corresponding grams (of size k or smaller) are observed preceding or following the index (preceding features are distinct from following ones); given a target string and index, we observe only preceding grams. Although it is possible to observe following grams in the target string in some settings, it is not possible in general (i.e., not when generating strings). We therefore adhere to this restriction for convenience and uniformity.

An example will make this clear. In Table 2 we are midway through the conversion of the source string "editing" into the target string "STRINGS". Below the two strings are those gram features which have non-zero value at the indicated cursors. The underbar character encodes on which side of the cursor a gram is observed. Note that an empty-gram feature, which always tests true, is also included, allowing us to experiment with 0-order models.

3.2 Illustrative Problems

To test the ability of the model to recover known edit affinities, we experimented with a simple artificial problem. Using a large list of English words, we define an edit affinity that is sensitive only to consonants. Specifically, the affinity between two words is the maximum number of consonant self-substitutions, with any substitutions involving the first five consonants counting for five normal substitutions. Thus, substituting 'b' for 'b' contributes 5 to the score, substituting 'z' for 'z' contributes 1, while operations other than self-subsitutions, and any operations involving vowels, contribute 0.

One epoch of training is conducted as follows. For each word s in the training set, we choose 10 other words from the set at random and sort these words according to both the true and estimated affinity. Let t be the string with highest true affinity; let t' (the decoy) be the string with highest estimated affinity. We performed 3-fold cross-validation on a collection of 33,432 words, in each fold training the model for 5 epochs.[2]

Our performance metric is *ranking accuracy*, the fraction of target string pairs to which the estimated ranking assigns the same order as the true one. During testing, for each source string, we sample at random 1000 other strings from the hold-out data, and count the fraction of all pairs ordered correctly according to this criterion.

A 0-order model successfully learns to rank strings according to this affinity with 99.3% accuracy, while ranking according to the unmodified Levenshtein distance yields 76.4%. Table 3 shows the 6 averaged weights with the highest magnitude from a model trained on one of the folds. In presenting weights, we follow the formatting convention *edit*:*feature*. Since the model in question is of order 0, all features in Table 3 are the "empty feature." Note how idempotent substitutions involving the 5 highly weighted consonants are weighted significantly higher than the remaining operations.

$\langle \mathtt{d}, \mathtt{d} \rangle$: $f_\emptyset$	61.1
$\langle \mathtt{c}, \mathtt{c} \rangle$: $f_\emptyset$	60.6
$\langle \mathtt{g}, \mathtt{g} \rangle$: $f_\emptyset$	60.3
$\langle \mathtt{b}, \mathtt{b} \rangle$: $f_\emptyset$	59.1
$\langle \mathtt{f}, \mathtt{f} \rangle$: $f_\emptyset$	57.0
$\langle \mathtt{t}, \mathtt{t} \rangle$: $f_\emptyset$	18.6

Table 3: Largest weights in a consonant-preserving edit affinity in which the first five consonants are given 5 times as much weight as others.

3.3 Rhyming

While the above problem illustrates the ability of the proposed algorithm to learn latent alignment affinities, it is expressible as a order-0 model. A somewhat more interesting problem is that of modeling groups of rhyming words. This problem is an instance of what we called the "classification" scenario in Section 2.3. Because English letters have long since lost their direct correspondence to phonemes, the problem of distinguishing rhyming English words is difficult for a knowledge-lean edit model. What's more, the importance of a letter is dependent on context; letters near the end of a word are more likely to be significant.

We derived groups of rhyming words from the CMU pronouncing dictionary (CMU, 1995), discarding any singleton groups. This yielded 21,396 words partitioned into 3,799 groups, ranging in size from 464 words (nation, location, etc.) down to 2. We then divided the words in this data set at random into three groups for cross-validation.

Training was conducted as follows. For each word in the training set, we selected at random up to 5 rhyming words and 5 non-rhyming words. These words were ranked according to affinity with the source word under the current model. Let t be the lowest scoring rhyming word, and let t' be the highest-scoring non-rhyming word.

[2]Here and in other experiments involving the edit model, the number of epochs was set arbitrarily, and *not* based on performance on a development set. Beyond the number of epochs required for convergence, we have not observed much sensitivity in test accuracy to the number of epochs.

Model	*Precision*
Levenshtein	0.126
Longest common suffix	0.130
PTEM, Order 0	0.505
PTEM, Order 3	0.790

Table 4: Micro-averaged break-even precision on the task of grouping rhyming English words.

For each word in the hold-out set, we scored and ranked all rhyming words in the same set, as well as enough non-rhyming words to total 1000. We then recorded the precision at the point in this ranking where recall and precision are most nearly equal. Our summary statistic is the micro-averaged break-even precision.

Table 4 presents the performance of the proposed model and compares it with two simple baselines. Not surprisingly, performance increases with increasing order. The simple heuristic approaches fare quite poorly by comparison, reflecting the subtlety of the problem.

3.4 Transcription

Our work was motivated by the problem of named entity transcription. Out-of-vocabulary (OOV) terms are a persistent problem in statistical machine translation. Often, such terms are the names of entities, which typically have low corpus frequencies. In translation, the appropriate handling of names is often to transcribe them, to render them idiomatically in the target language in a way that preserves, as much as possible, their phonetic structure. Even when an OOV term is not a name, transcribing it preserves information that would otherwise be discarded, leaving open the possibility that downstream applications will be able to make use of it.

The state of the art in name transcription involves some form of generative model, sometimes in combination with additional heuristics. The generative component may involve explicitly modeling phonetics. For example, Knight and Graehl (1998) employ cascaded probabilistic finite-state transducers, one of the stages modeling the orthographic-to-phonetic mapping. Subsequently, Al-Onaizan and Knight (2002) find they can boost performance by combining a phonetically-informed model with one trained only on orthographic correspondences. Huang et al. (2004), construct a probabilistic Chinese-English edit model as part of a larger alignment solution, setting edit weights in a heuristic bootstrapped procedure.

Task	*Train*	*Dev*	*Eval*	*ELen*	*FLen*
A-E	8084	1000	1000	6.5	4.9
M-E	2000	430	1557	16.3	23.0

Table 5: Characteristics of the two transcription data sets, Arabic-English (A-E) and Mandarin-English (M-E), including number of training, development, and evaluation pairs (*Train*, *Dev*, and *Eval*), and mean length in characters of English and foreign strings (*ELen* and *FLen*).

In rendering unfamiliar written Arabic words or phrases in English, it is generally impossible to achieve perfect performance, because many sounds, such as short vowels, diphthong markers, and doubled consonants, are conventionally not written in Arabic. We calculate from our experimental datasets that approximately 25% of the characters in the English output must be inferred. Thus, a character error rate of 25% can be achieved through simple transliteration.

3.4.1 Transcribing names

We experimented with a list of 10,084 personal names distributed by the Linguistic Data Consortium (LDC). Each entry in the database includes an arabic name in transliterated ASCII (SATTS method) and its English rendering. The Arabic names appear as they would in conventional written Arabic, i.e., lacking short vowels and other diacritics. We randomly segregated 1000 entries for evaluation and used the rest for training. The *A-E* row in Table 5 summarizes some of this data set's characteristics.

We trained the edit model as follows. For each training pair the indicated English rendering constitutes our true target (t), and we use the current model to generate an alternate string (t'), updating the model in the event t' yields a higher score than t. This was repeated for 10 epochs. We experimented with a model of order 3.

Under this methodology, we observed a 1-best ac-

$\langle \texttt{p}, \texttt{h} \rangle$:	$f_{t,a\_}$	38.5
$\langle \texttt{p}, \texttt{t} \rangle$:	$f_{t,a\_}$	30.8
$\langle \texttt{p}, \texttt{h} \rangle$:	$f_{t,ya\_}$	11.8
$\langle \texttt{p}, \texttt{t} \rangle$:	$f_{s,\_p<e>}$	-8.6
$\langle \texttt{p}, \texttt{h} \rangle$:	$f_{t,rya\_}$	-12.1
$\langle \texttt{p}, \texttt{t} \rangle$:	$f_{t,uba\_}$	-14.4

Table 6: Some of the weights governing the handling of the *tah marbouta* (ة) in an order-3 Arabic-English location name transcription model. Buckwalter encoding of Arabic characters is used here for purposes of display. The symbol "<e>" represents end of string.

curacy of 0.552. It is difficult to characterize the strength of this result relative to those reported in the literature. Al-Onaizan and Knight (2002) report a 1-best accuracy of 0.199 on a corpus of Arabic *person* names (but an accuracy of 0.634 on *English* names), using a "spelling-based" model, i.e., a model which has no access to phonetic information. However, the details of their experiment and model differ from ours in a number of respects.

It is interesting to see how a learned edit model handles ambiguous letters. Table 6 shows the weights of some of the features governing the handling of the character ة (*tah marbouta*) from experiments with Arabic place names. This character, which represents the "t" sound, typically appears at the end of words. It is generally silent, but is spoken in certain grammatical constructions. In its silent form, it is typically transcribed "ah" (or "a"); in its spoken form, it is transcribed "at". The weights in the table reflect this ambiguity and illustrate some of the criteria by which the model chooses the appropriate transcription. For example, the negative weight on the feature $f_{s,\_p<e>}$ inhibits the production of "t" at the end of a phrase, where "h" is almost always more appropriate. Similarly, "h" is more common following "ya" in the target string (often as part of the larger suffix "iyah"). However, the preceding context "rya" is usually observed in the word "qaryat", meaning "village" as in "the village of ..." In this grammatical usage, the tah marbouta is spoken and therefore rendered with a "t". Consequently, the corresponding weight in the "h" interpretation is inhibitory.

The Al-Onaizan and Knight spelling model can be regarded as a statistical machine translation (SMT) system which translates source language characters to target language characters in the absence of phonetic information. For comparison with state of the art, we used the RWTH phrase-based SMT system (Zens et al., 2005) to build an Arabic-to-English transliteration system. This system frames the transcription problem as follows. We are given a sequence of source language characters s_1^m representing a name, which is to be translated into a sequence of target language characters t_1^n. Among all possible target language character sequences, we will choose the character sequence with the highest probability:

$$\hat{t}_1^{\hat{n}} = \operatorname*{argmax}_{n, t_1^n} \{ \Pr(t_1^n | s_1^m) \} \tag{3}$$

The posterior probability $\Pr(t_1^n | s_1^m)$ is modeled directly using a log-linear combination of several models (Och and Ney, 2002), including a character-based phrase translation model, a character-based lexicon model, a 4-gram character sequence model, a character penalty and a phrase penalty. The first two models are used for both directions: Arabic to English and English to Arabic. We do not use any reordering model because the target character sequence is always monotone with respect to the source character sequence. More details about the baseline system can be found in (Zens et al., 2005).

We remark in passing that while the perceptron-based edit model is a general algorithm for learning sequence alignments using simple features, the above SMT approach combines several models, some of which have been the subject of research in the fields of speech recognition and machine translation for several years. Furthermore, we made an effort to optimize the performance of the SMT approach on the tasks presented here.

Table 7 compares this system with the edit model. The difference between the 1-best accuracies of the two systems is significant at the 95% level, using the bootstrap for testing. However, we can improve on both systems by combining them. We segregated 1000 training documents to form a development set, and used it to learn linear combination coefficients over our two systems, resulting in a combined system that scored 0.588 on the evaluation set—a sta-

Model	*1best*	*5best*
SMT	0.528	0.824
PTEM, Order 3	0.552	0.803
Linear combination	0.588	0.850

Table 7: 1-best and 5-best transcription accuracies. The successive improvements in 1-best accuracy are significant at the 95% confidence level.

tistically significant improvement over both systems at the 95% confidence level.

3.4.2 Ranking transcriptions

In some applications, instead of transcribing a name in one language into another, it is enough just to rank candidate transcriptions. For example, we may be in possession of comparable corpora in two languages and the means to identify named entities in each. If we can rank the likely transcriptions of a name, we may be able to align a large portion of the transliterated named entities, potentially extending the coverage of our machine translation system, which will typically have been developed using a smaller parallel corpus. This idea is at the heart of several recent attempts to improve the handling of named entities in machine translation (Huang et al., 2004; Lee and Chang, 2003). A core component of all such approaches is a generative model similar in structure to the "spelling" model proposed by Al-Onaizan and Knight.

When ranking is the objective, we can adopt a training procedure that is much less expensive than the one used for generation. Let t be the correct transcription for a source string (s). Sample some number of strings at random (200 in the following experiments) from among the transcriptions in the training set of strings other than s. Let t' be the string having highest affinity with s, updating the model, as usual, if t' scores higher than t.

In addition to the Arabic-English corpus, we also experiment with a corpus distributed by the LDC of full English names paired with their Mandarin spelling. The *M-E* row of Table 5 summarizes characteristics of this data set. Because we are interested in an approximate comparison with similar experiments in the literature, we selected at random 2430 for training and 1557 for evaluation, which are the data sizes used by Lee and Chang (2003) for their experiments. In these experiments, the Chinese names are represented as space-separated pinyin without tonal markers.

Note that this problem is probably harder than the Arabic one, for several reasons. For one thing, the letters in a Mandarin transcription of a foreign name represent syllables, leading to a somewhat lossier rendering of foreign names in Mandarin than in Arabic. On a more practical level, this data set is noisier, occasionally containing character sequences in one string for which corresponding characters are lacking from its paired string. On the other hand, the Mandarin problem contains full names, rather than name components, which provides more context for ranking.

We trained the edit model on both data sets using both the sampling procedure outlined above and the self-generation training regime, in each case for 20 epochs, producing models of orders from 1 to 3. However, we found that the efficiency of the phrase-based SMT system described in the previous section would be limited for this task, mainly due to two reasons: the character-based phrase models due to possible unseen phrases in an evaluation corpus, and the character sequence model as all candidate transcriptions confidently belong to the target language. Therefore, to make the phrase-based SMT system robust against data sparseness for the ranking task, we also make use of the IBM Model 4 (Brown et al., 1993) in both directions. The experiments show that IBM Model 4 is a reliable model for the ranking task. For each evaluation pair, we then ranked all available evaluation transcriptions, recording where in this list the true transcription fell.

Table 8 compares the various models, showing the fraction of cases for which the true transcription was ranked highest, and its mean reciprocal rank (MRR). Both the phrase-based SMT model and the edit model perform well on this task. While the best configuration of PTEM out-performs the best SMT model, the differences are not significant at the 95% confidence level. However, compare these performance scores to those returned by the system of Lee and Chang (2003), who reported a peak MRR of 0.82 in similar experiments involving data different from ours.

The PTEM rows in the table are separated into

Model	C-E Task		A-E Task	
	ACC	MRR	ACC	MRR
SMT	0.795	0.797	0.982	0.985
SMT w/o LM	0.797	0.798	0.983	0.985
IBM_4	0.961	0.971	0.978	0.987
SMT + IBM_4	0.971	0.977	0.991	0.994
PTEM*G*, Ord. 1	0.843	0.877	0.959	0.975
PTEM*G*, Ord. 2	0.970	0.978	0.968	0.980
PTEM*G*, Ord. 3	0.975	0.982	0.971	0.983
PTEM*R*, Ord. 1	0.961	0.973	0.992	0.995
PTEM*R*, Ord. 2	0.960	0.972	0.989	0.993
PTEM*R*, Ord. 3	0.960	0.972	0.989	0.994

Table 8: Performance on two transcription ranking tasks, showing fraction of cases in which the correct transcription was ranked highest, accuracy (ACC) and mean reciprocal rank of the correct transcription (MRR).

those in which the model was trained using the same procedure as for generation (PTEM*G*), and those in which the quicker ranking-specific training regime was used (PTEM*R*). The comparison is interesting, inasmuch it does not support the conclusion that one regime is uniformly superior to the other. While generation regime yields the best performance on Arabic (using a high-order model), the ranking regime scores best on Mandarin (with a low-order model). When training a model to generate, it seems clear that more context in the form of larger n-grams is beneficial. This is particularly true for Mandarin, where an order-1 model probably does not have the capacity to generate plausible decoys.

4 Discussion

This paper is not the first to show that perceptron training can be used in the solution of problems involving transduction. Both Liang, et al (2006), and Tillmann and Zhang (2006) report on effective machine translation (MT) models involving large numbers of features with discriminatively trained weights. The training of these models is an instance of the "Generation" scenario outlined in Section 2.3. However, because machine translation is a more challenging problem than name transcription (larger vocabularies, higher levels of ambiguity, non-monotonic transduction, etc.), our general-purpose approach to generation training may be intractable for MT. Instead, much of the focus of these papers are the heuristics that are required in order to train such a model in this fashion, including feature selection using external resources (phrase tables), staged training, and generating to BLEU-maximal sequences, rather than the reference target.

Klementiev and Roth (2006) explore the use of a perceptron-based ranking model for the purpose of finding name transliterations across comparable corpora. They do not calculate an explicit alignment between strings. Instead, they decompose a string pair into a collection of features derived from character n-grams heuristically paired based on their locations in the respective strings. Thus, Klementiev and Roth, in common with the two MT approaches described above, carefully control the features used by the perceptron. In contrast to these approaches, our algorithm discovers *latent* alignments, essentially selecting those features necessary for good performance on the task at hand.

As noted in the introduction, several previous papers have proposed general, discriminatively trained sequence alignment models, as alternatives to the generative model proposed by Ristad and Yianilos. McCallum, et al. (2005), propose a conditional random field for sequence alignment, designed for the important problem of duplicate detection and information integration. Comprising two sub-models, one for matching strings and one for non-matching, the model is trained on sequence pairs explicitly labeled "match" or "non-match," and some care is apparently needed in selecting appropriate non-matching strings. It is therefore unclear how this model would be extended to problems involving ranking or generation.

Joachims (2003) proposes SVM-align, a sequence alignment model similar in structure to that described here, but which sets weights through direct numerical optimization. Training involves exposing the model to sequence pairs, along with the correct alignment and some number of "decoy" sequences. The reliance on an explicit alignment and hand-chosen decoys yields a somewhat less flexible solution than that presented here. It is not clear whether these features of the training regime are indispensable, or whether they might be generalized to increase the approach's scope. Note that where directly maximizing the margin is feasible, it has been shown empirically to be superior to perceptron training (Altun et al., 2003).

Parker et al. (2006), propose to align sequences by gradient tree boosting. This approach has the attractive characteristic that it supports a factored representation of edits (a characteristic it shares with McCallum et al.). Although this paper does not evaluate the method on any problems from computational linguistics (the central problem is musical information retrieval), gradient tree boosting has been shown to be an effective technique for other sorts of sequence modeling drawn from computational linguistics (Dietterich et al., 2004).

5 Conclusion

Motivated by the problem of Arabic-English transcription of names, we adapted recent work in perceptron learning for sequence labeling to the problem of sequence alignment. The resulting algorithm shows clear promise not only for transcription, but also for ranking of transcriptions and structural classification. We believe this versatility will lead to other successful applications of the idea, both within computational linguistics and in other fields involving sequential learning.

Acknowledgment of support

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA/IPTO) under Contract HR0011-06-C-0023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA).

References

Y. Al-Onaizan and K. Knight. 2002. Machine transliteration of names in Arabic text. In *Proceedings of the ACL-02 workshop on computational approaches to semitic languages*.

Y. Altun, I. Tsochantaridis, and T. Hofmann. 2003. Hidden Markov support vector machines. In *Proceedings of ICML-2003*.

M. Bilenko and R. Mooney. 2003. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of KDD-2003*.

P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2), June.

CMU. 1995. The CMU pronouncing dictionary. http://www.speech.cs.cmu.edu/cgi-bin/cmudict. Version 0.6.

M. Collins. 2002. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proceedings of EMNLP-2002*.

T. Dietterich, A. Ashenfelter, and Y. Bulatov. 2004. Training conditional random fields via gradient tree boosting. In *Proceedings of ICML-2004*.

F. Huang, S. Vogel, and A. Waibel. 2004. Improving named entity translation combining phonetic and semantic similarities. In *Proceedings of HLT-NAACL 2004*.

T. Joachims. 2003. Learning to align sequences: a maximum-margin approach. Technical report, Cornell University.

A. Klementiev and D. Roth. 2006. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *Proceedings of Coling/ACL 2006*.

K. Knight and J. Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4).

C.-J. Lee and J.S. Chang. 2003. Acquisition of English-Chinese transliterated word pairs from parallel-aligned texts using a statistical machine transliteration model. In *Proceedings of the HLT-NAACL 2003 Workshop on Building and Using Parallel Texts*.

P. Liang, A. Bouchard-Côté, D. Klein, and B. Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proceedings of COLING 2006/ACL 2006*.

A. McCallum, K. Bellare, and F. Pereira. 2005. A conditional random field for discriminatively-trained finite-state string edit distance. In *Proceedings of UAI-2005*.

S.B. Needleman and C.D. Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48.

F.J. Och and H. Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *ACL02*, pages 295–302, Philadelphia, PA, July.

C. Parker, A. Fern, and P Tadepalli. 2006. Gradient boosting for sequence alignment. In *Proceedings of AAAI-2006*.

E.S. Ristad and P.N. Yianilos. 1998. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20.

C. Tillmann and T. Zhang. 2006. A discriminative global training algorithm for statistical MT. In *Proceedings of Coling/ACL 2006*.

R. Zens, O. Bender, S. Hasan, S. Khadivi, E. Matusov, J. Xu, Y. Zhang, and H. Ney. 2005. The RWTH phrase-based statistical machine translation system. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*, pages 155–162, Pittsburgh, PA, October.