Loss-Sensitive Discriminative Training of Machine Transliteration Models

Kedar Bellare
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003, USA
kedarb@cs.umass.edu

Koby Crammer
Department of Computer Science
University of Pennsylvania
Philadelphia, PA 19104, USA
crammer@cis.upenn.edu

Dayne Freitag
SRI International
San Diego, CA 92130, USA
dayne.freitag@sri.com

Abstract

In machine transliteration we transcribe a name across languages while maintaining its phonetic information. In this paper, we present a novel sequence transduction algorithm for the problem of machine transliteration. Our model is discriminatively trained by the MIRA algorithm, which improves the traditional Perceptron training in three ways: (1) It allows us to consider k-best transliterations instead of the best one. (2) It is trained based on the ranking of these transliterations according to user-specified loss function (Levenshtein edit distance). (3) It enables the user to tune a built-in parameter to cope with noisy non-separable data during training. On an Arabic-English name transliteration task, our model achieves a relative error reduction of 2.2% over a perceptron-based model with similar features, and an error reduction of 7.2% over a statistical machine translation model with more complex features.

1 Introduction and Related Work

Proper names and other technical terms are frequently encountered in natural language text. Both machine translation (Knight and Graehl, 1997) and cross-language information retrieval (Jeong et al., 1999; Virga and Khudanpur, 2003; Abdul-Jaleel and Larkey, 2003) can benefit by explicitly translating such words from one language into another. This approach is decidedly better than treating them uniformly as out-of-vocabulary tokens. The goal of machine *transliteration* is to translate words between alphabets of different languages such that they are phonetically equivalent.

Given a source language sequence $\mathbf{f} = f_1 f_2 \dots f_m$ from an alphabet $\mathcal{F}$, we want to produce a target language sequence $\mathbf{e} = e_1 e_2 \dots e_n$ in the alphabet $\mathcal{E}$ such that it maximizes some score function $s(\mathbf{e}, \mathbf{f})$,

$$\mathbf{e} = \arg\max_{\mathbf{e}'} s(\mathbf{e}', \mathbf{f}).$$

Virga and Khudanpur (2003) model this scoring function using a separate *translation* and *language* model, that is, $s(\mathbf{e}, \mathbf{f}) = Pr(\mathbf{f}|\mathbf{e})Pr(\mathbf{e})$. In constrast, Al-Onaizan and Knight (2002) directly model the translation probability $Pr(\mathbf{e}|\mathbf{f})$ using a log-linear combination of several individually trained phrase and character-based models. Others have treated transliteration as a phrase-based transduction (Sherif and Kondrak, 2007). All these approaches are adaptations of statistical models for machine translation (Brown et al., 1994). In general, the parameters of the scoring function in such approaches are trained generatively and do not utilize complex features of the input sequence pairs.

Recently, there has been interest in applying discriminatively-trained sequence alignment models to many real-world problems. McCallum et al. (2005) train a conditional random field model to discriminate between matching and non-matching string pairs treating alignments as latent. Learning accurate alignments in this model requires finding "close" non-match pairs which can be a challenge. A similar conditional latent-variable model has been applied to the task of lemmatization and generation of morphological forms (Dreyer et al., 2008).

Zelenko and Aone (2006) model transliteration as a structured prediction problem where the letter e_i is predicted using local and global features derived from $e_1 e_2 \ldots e_{i-1}$ and $\mathbf{f}$. Bergsma and Kondrak (2007) address cognate identification by training a SVM classification model using phrase-based features obtained from a Levenshtein alignment. Both these models do not learn alignments that is needed to obtain high performance on transliteration tasks. Freitag and Khadivi (2007) describe a discriminatively trained sequence alignment model based on averaged perceptron, which is closely related to the method proposed in this paper.

Our approach improves over previous directions in two ways. First, our system produces better k-best transliterations than related approaches by training on multiple hypotheses ranked according to a user-specified loss function (Levenshtein edit distance). Hence, our method achieves a 19.2% error reduction in 5-best performance over a baseline only trained with 1-best transliterations. This is especially helpful when machine transliteration is part of a larger machine translation or information retrieval pipeline since additional sentence context can be used to choose the best among top-K transliterations. Second, our training procedure accounts for noise and non-separability in the data. Therefore, our transliteration system would work well in cases where person names were misspelled or in cases in which a single name had many reasonable translations in the foreign language.

The training algorithm we propose in this paper is based on the K-best MIRA algorithm which has been used earlier in structured prediction problems (McDonald et al., 2005a; McDonald et al., 2005b). Our results demonstrate a significant improvement in accuracy of 7.2% over a statistical machine translation (SMT) system (Zens et al., 2005) and of 2.2% over a perceptron-based edit model (Freitag and Khadivi, 2007).

2 Sequence Alignment Model

Let $\mathbf{e} = e_1 e_2 \ldots e_n$ and $\mathbf{f} = f_1 f_2 \ldots f_m$ be sequences from the target alphabet $\mathcal{E}$ and source alphabet $\mathcal{F}$ respectively. Let $\mathbf{a} = a_1 a_2 \ldots a_l$ be a sequence of alignment operations needed to convert $\mathbf{f}$ into $\mathbf{e}$. Each alignment operation either appends a letter to the end of the source sequence, the target sequence or both sequences. Hence, it is a member of the cross-product $a_k \in \mathcal{E} \cup \{\epsilon\} \times \mathcal{F} \cup \{\epsilon\} \backslash \{(\epsilon, \epsilon)\}$, where ϵ is the null character symbol. Let $\mathbf{a}_1^k = a_1 a_2 \ldots a_k$ denote the sequence of first k alignment operations. Similarly $\mathbf{e}_1^k$ and $\mathbf{f}_1^k$ are prefixes of $\mathbf{e}$ and $\mathbf{f}$ of length k.

We define the scoring function between a word and its transliteration to be the a maximum over all possible alignment sequences $\mathbf{a}$,

$$s(\mathbf{e}, \mathbf{f}) = \max_{\mathbf{a}} s(\mathbf{a}, \mathbf{e}, \mathbf{f}) \ ,$$

where the score of a specific alignment $\mathbf{a}$ between two words is given by a linear relation,

$$s(\mathbf{a}, \mathbf{e}, \mathbf{f}) = \mathbf{w} \cdot \Phi(\mathbf{a}, \mathbf{e}, \mathbf{f}),$$

for a parameter vector $\mathbf{w}$ and a feature vector $\Phi(\mathbf{a}, \mathbf{e}, \mathbf{f})$. Furthermore, let $\Phi(\mathbf{a}, \mathbf{e}, \mathbf{f}) = \sum_{k=1}^{l} \phi(a_k, \mathbf{e}, i, \mathbf{f}, j)$ be the sum of feature vectors associated with individual alignment operations. Here i, j are positions in sequences $\mathbf{e}, \mathbf{f}$ after performing operations $\mathbf{a}_1^k$. For fixed sequences $\mathbf{e}$ and $\mathbf{f}$ the function $s(\mathbf{e}, \mathbf{f})$ can be efficiently computed using a dynamic programming algorithm,

$$s(\mathbf{e}_1^i, \mathbf{f}_1^j) = \max \begin{cases} s(\mathbf{e}_1^{i-1}, \mathbf{f}_1^j) + \mathbf{w} \cdot \phi(\langle e_i, \epsilon \rangle, \mathbf{e}, i, \mathbf{f}, j) \\ s(\mathbf{e}_1^i, \mathbf{f}_1^{j-1}) + \mathbf{w} \cdot \phi(\langle \epsilon, f_j \rangle, \mathbf{e}, i, \mathbf{f}, j) \\ s(\mathbf{e}_1^{i-1}, \mathbf{f}_1^{j-1}) + \mathbf{w} \cdot \phi(\langle e_i, f_j \rangle, \mathbf{e}, i, \mathbf{f}, j). \end{cases} \quad (1)$$

Given a source sequence $\mathbf{f}$ computing the best scoring target sequence $\mathbf{e} = \arg\max_{\mathbf{e}'} s(\mathbf{e}', \mathbf{f})$ among all possible sequences $\mathcal{E}^*$ requires a beam search procedure (Freitag and Khadivi, 2007). This procedure can also be used to produce K-best target sequences $\{\mathbf{e}'_1, \mathbf{e}'_2, \ldots, \mathbf{e}'_K\}$ such that $s(\mathbf{e}'_1, \mathbf{f}) \geq s(\mathbf{e}'_2, \mathbf{f}) \geq \ldots \geq s(\mathbf{e}'_K, \mathbf{f})$.

In this paper, we employ the same features as those used by Freitag and Khadivi (2007). All local feature functions $\phi(a_k, \mathbf{e}, i, \mathbf{f}, j)$ are conjunctions of the alignment operation a_k and forward or backward-looking character m-grams in sequences $\mathbf{e}$ and $\mathbf{f}$ at positions i and j respectively. For the source sequence $\mathbf{f}$ both forward and backward-looking m-gram features are included. We restrict the m-gram features in our target sequence $\mathbf{e}$ to only

be backward-looking since we do not have access to forward-looking m-grams during beam-search. An order M model is one that uses m-gram features where $m = 0, 1, \ldots M$.

Our training algorithm takes as input a data set $\mathcal{D}$ of source-target transliteration pairs and outputs a parameter vector $\mathbf{u}$. The algorithm pseudo-code appears in Fig. (1). In the algorithm, the function $\mathcal{L}(\mathbf{e}', \mathbf{e})$ defines a loss incurred by predicting $\mathbf{e}'$ instead of $\mathbf{e}$. In most structured prediction problems, the targets are of equal length and in such cases the Hamming loss function can be used. However, in our case the targets may differ in terms of length and thus we use the Levenshtein edit distance (Levenshtein, 1966) with unit costs for insertions, deletions and substitutions. Since the targets are both in the same alphabet $\mathcal{E}$ this loss function is well-defined. The user also supplies three paramters: (1) T - the number of training iterations (2) K - the number of best target hypotheses used (3) C - a complexity parameter. A low C is useful if the data is non-separable and noisy.

The final parameter vector $\mathbf{u}$ returned by the algorithm is the average of the intermediate parameter vectors produced during training. We find that averaging helps to improve performance. At test time, we use the beam search procedure to produce K-best hypotheses using the parameter vector $\mathbf{u}$.

3 Experimental Results

We apply our model to the real-world Arabic-English name transliteration task on a data set of 10,084 Arabic names from the LDC. The data set consists of Arabic names in an ASCII-based alphabet and its English rendering. Table 1 shows a few examples of Arabic-English pairs in our data set. We use the same training/development/testing (8084/1000/1000) set as the one used in a previous benchmark study (Freitag and Khadivi, 2007). The development and testing data were obtained by randomly removing entries from the training data. The absence of short vowels (e.g. "a" in ⟨NB"I, nab'i⟩), doubled consonants (e.g. "ww" in ⟨FWAL, fawwal⟩) and other diacritics in Arabic make the transliteration a hard problem. Therefore, it is hard to achieve perfect accuracy on this data set.

For training, we set $K = 20$ best hypotheses and

Input parameters

Training Data	$\mathcal{D}$
Complexity parameter	$C > 0$
Number of epochs	T

Initialize $\mathbf{w}_0 = \mathbf{0}$ (zero vector) ; $\tau = 0$; $\mathbf{u} = \mathbf{0}$

Repeat T times:

For Each $(\mathbf{e}, \mathbf{f}) \in \mathcal{D}$:

1. $\mathbf{a} = \arg\max_{\hat{\mathbf{a}}} \mathbf{w}_\tau \cdot \Phi(\hat{\mathbf{a}}, \mathbf{e}, \mathbf{f})$ (Find best scoring alignment between $\mathbf{e}$ and $\mathbf{f}$ using dynamic programming)
2. Generate a list of K-best target hypotheses $\{\mathbf{e}'_1, \mathbf{e}'_2, \ldots, \mathbf{e}'_K\}$ given the current parameters $\mathbf{w}_\tau$. Let the corresponding alignments for the targets be $\{\mathbf{a}'_1, \mathbf{a}'_2, \ldots, \mathbf{a}'_K\}$.
3. Set $\mathbf{w}_{\tau+1}$ to be the solution of :

$$\min_{\mathbf{w}} \tfrac{1}{2}||\mathbf{w} - \mathbf{w}_\tau||^2 + C\sum_{k=1}^{K} \xi_k$$
subject to (for $k = 1 \ldots K$) :
$$\mathbf{w} \cdot (\Phi(\mathbf{a}, \mathbf{e}, \mathbf{f}) - \Phi(\mathbf{a}'_k, \mathbf{e}'_k, \mathbf{f})) \geq \mathcal{L}(\mathbf{e}, \mathbf{e}'_k) - \xi_k$$
$$\xi_k \geq 0$$

4. $\mathbf{u} \leftarrow \mathbf{u} + \mathbf{w}_{\tau+1}$
5. $\tau \leftarrow \tau + 1$

Output Scoring function $s(\mathbf{a}, \mathbf{e}, \mathbf{f}) = \mathbf{u} \cdot \Phi(\mathbf{a}, \mathbf{e}, \mathbf{f})$

Figure 1: The k-best MIRA algorithm for discriminative learning of transliterations.

Arabic	English
NB"I	nab'i
HNBLI	hanbali
FRIFI	furayfi
MLKIAN	malikian
BI;ANT	bizant
FWAL	fawwal
OALDAWI	khalidawi
BUWUI	battuti
H;?	hazzah

Table 1: Examples of Arabic names in the ASCII alphabet and their English transliterations.

$C = 1.0$ and run the algorithm for $T = 10$ epochs. To evaluate our algorithm, we generate 1-best (or 5-best) hypotheses using the beam search procedure and measure accuracy as the percentage of instances in which the target sequence $\mathbf{e}$ is one of the 1-best (or 5-best) targets. The input features are based on character m-grams for $m = 1, 2, 3$. Unlike previ-

ous generative transliteration models, no additional language model feature is used.

We compare our model against a state-of-the-art statistical machine translation (SMT) system (Zens et al., 2005) and an averaged perceptron edit model (PTEM) with identical features (Freitag and Khadivi, 2007). The SMT system directly models the posterior probability $Pr(\mathbf{e}|\mathbf{f})$ using a log-linear combination of several sub-models: a character-based phrase translation model, a character-based lexicon model, a character penalty and a phrase penalty. In the PTEM model, the update rule only considers the best target sequence and modifies the parameters $\mathbf{w}_{\tau+1} = \mathbf{w}_\tau + \Phi(\mathbf{a}, \mathbf{e}, \mathbf{f}) - \Phi(\mathbf{a}', \mathbf{e}', \mathbf{f})$ if the score $s(\mathbf{e}', f) \geq s(\mathbf{e}, f)$.

Model (*train+dev*)	*1-best*	*5-best*
SMT	0.528	0.824
PTEM	0.552	0.803
MIRA	0.562	0.841

Table 2: The 1-best and 5-best accuracy of different models on the Arabic-English transliteration task. At 95% confidence level, MIRA/PTEM outperform the SMT model in 1-best accuracy and MIRA outperforms PTEM/SMT in 5-best accuracy.

Table 2 shows the *1-best* and *5-best* accuracy of each model trained on the combined *train+dev* data set. All the models are evaluated on the same *test* set. Both MIRA and PTEM algorithms outperform the SMT model in terms of *1-best accuracy*. The differences in accuracy are significant at 95% confidence level, using the bootstrapping method for hypothesis testing. The difference in *1-best* performance of MIRA and PTEM is not significant. At *5-best*, the MIRA model outperforms both SMT and PTEM model. We conjecture that using the problem-specific Levenshtein loss function helps filter bad target sequences from the K-best outputs during training.

In a second experiment we studied the effect of changing C on the performance of the algorithm. We ran the algorithm with the above settings, except varying the value of the complexity parameter to one of 7 values in the range $C = 0.00001, 0.0001, \ldots, 0.1, 1.0$, training only using the *train* set, and evaluating the resulting model on the *test* set. The results are summarized in Table 3. The entry marked with a star $*$ indicates the model that achieved the best performance on the *dev* set for a particular choice of evaluation measure (1-best or 5-best). We find that changing C does have an effect on model performance. As the value of C decreases, the performance at lower ranks improves: $C = 0.01$ is good for 5-best accuracy and $C = 0.001$ for 20-best accuracy (not in table). As C is further reduced, a greater number of iterations are needed to converge. In our model, where the alignments are not observed but inferred during training, we find that making small incremental updates makes our algorithm more robust. Indeed, setting $C = 0.01$ and training on the *train+dev* set improves 5-best performance of our model from 0.841 to 0.861. Hence, the choice of C is important.

Model (*train*)	*1-best*	*5-best*
$C = 1.0$	0.545^*	0.832
$C = 0.5$	0.548^*	0.83
$C = 0.2$	0.549^*	0.834
$C = 0.01$	0.545	0.852^*
$C = 0.001$	0.518	0.843
$C = 0.0001$	0.482	0.798
$C = 0.00001$	0.476	0.798

Table 3: The effect of varying model parameter C on *1,5-best* accuracy on the *test* set. All the models are trained with Levenshtein loss and 20-best targets. The superscript $*$ indicates the models that achieved the greatest performance on the *dev* set for a particular column.

4 Conclusions and Future Work

We have shown a significant improvement in accuracy over state-of-the-art transliteration models by taking into consideration the ranking of multiple hypotheses (top-K) by Levenshtein distance, and making the training algorithm robust to noisy non-separable data. Our model does consistently well at high ($K = 1$) *and* low ranks ($K = 5$), and can therefore be used in isolation or in a pipelined system (e.g. machine translation or cross-language information retrieval) to achieve better performance. In a pipeline system, more features of names around proper nouns and previous mentions of the name can be used to improve scoring of K-best outputs.

In our experiments, the Levenshtein loss function uses only unit costs for edit operations and is not specifically tuned towards our application. In future work, we may imagine penalizing insertions and deletions higher than substitutions and other non-uniform schemes for better transliteration performance. Our K-best framework can also be easily extended to cases where one name has multiple foreign translations that are equally likely.

References

Nasreen Abdul-Jaleel and Leah S. Larkey. 2003. Statistical transliteration for English-Arabic cross language information retrieval. In *CIKM '03*, pages 139–146, New York, NY, USA. ACM.

Yaser Al-Onaizan and Kevin Knight. 2002. Machine transliteration of names in arabic text. In *Proceedings of the ACL-02 Workshop on Computational Approaches to Semitic Languages*, pages 1–13.

Shane Bergsma and Greg Kondrak. 2007. Alignment-based discriminative string similarity. In *ACL*, pages 656–663, June.

Peter F. Brown, Stephen Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1994. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Markus Dreyer, Jason Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1080–1089, Honolulu, Hawaii, October. Association for Computational Linguistics.

Dayne Freitag and Shahram Khadivi. 2007. A sequence alignment model based on the averaged perceptron. In *EMNLP-CoNLL*, pages 238–247.

K.S. Jeong, S. H. Myaeng, J.S. Lee, and K.-S. Choi. 1999. Automatic identification and back-transliteration of foreign words for information retrieval. *Information Processing and Management*, 35:523–540.

Kevin Knight and Jonathan Graehl. 1997. Machine transliteration. In Philip R. Cohen and Wolfgang Wahlster, editors, *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 128–135, Somerset, New Jersey. Association for Computational Linguistics.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710.

Andrew McCallum, Kedar Bellare, and Fernando Pereira. 2005. A conditional random field for discriminatively-trained finite-state string edit distance. In *UAI*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Flexible text segmentation with structured multilabel classification. In *HLT-EMNLP*, pages 987–994, Vancouver, BC, Canada, October. Association for Computational Linguistics.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005b. Online large-margin training of dependency parsers. In *ACL*, pages 91–98, Ann Arbor, Michigan, June.

Tarek Sherif and Grzegorz Kondrak. 2007. Substring-based transliteration. In *ACL*, pages 944–951, Prague, Czech Republic, June. Association for Computational Linguistics.

Paola Virga and Sanjeev Khudanpur. 2003. Transliteration of proper names in cross-lingual information retrieval. In *Proceedings of the ACL 2003 workshop on Multilingual and Mixed-language Named Entity Recognition*, pages 57–64, Morristown, NJ, USA. Association for Computational Linguistics.

Dmitry Zelenko and Chinatsu Aone. 2006. Discriminative methods for transliteration. In *EMNLP*, pages 612–617, Sydney, Australia, July. Association for Computational Linguistics.

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)*, Pittsburgh, PA, USA.