Name Transliteration with Bidirectional Perceptron Edit Models

Dayne Freitag
SRI International
freitag@ai.sri.com

Zhiqiang (John) Wang
SRI International
johnwang@ai.sri.com

Abstract

We report on our efforts as part of the shared task on the NEWS 2009 Machine Transliteration Shared Task. We applied an orthographic perceptron character edit model that we have used previously for name transliteration, enhancing it in two ways: by ranking possible transliterations according to the sum of their scores according to two models, one trained to generate left-to-right, and one right-to-left; and by constraining generated strings to be consistent with character bigrams observed in the respective language's training data. Our poor showing in the official evaluation was due to a bug in the script used to produce competition-compliant output. Subsequent evaluation shows that our approach yielded comparatively strong performance on all alphabetic language pairs we attempted.

1 Introduction

While transliteration is a much simpler problem than another linguistic transduction problem, language translation, it is rarely trivial. At least three phenomena complicate the automatic transliteration between two languages using different scripts—differing phoneme sets, lossy orthography, and non-alphabetic orthographies (e.g., syllabaries).

For most language pairs, these difficulties stand in the way of a rule-based treatment of the problem. For this reason, many machine learning approaches to the problem have been proposed. We can draw a rough distinction between learning approaches that attempt to model the phonetics of a transliteration problem explicitly, and those that treat the problem as simply one of orthographic transduction, leaving it to the learning algorithm to acquire phonetic distinctions directly from orthographic features of the training data. For example, Knight and Graehl (1998) address the problem through cascaded finite state transducers, with explicit representations of the phonetics. Subsequently, Al-Onaizan and Knight (2002) realize improvements by adding a "spelling" (i.e., orthographic) model. There has been an increasing emphasis on purely orthographic models, probably because they require less detailed domain knowledge (e.g., (Lee and Chang, 2003)).

2 Approach

The approach we explored as part of the NEWS 2009 Machine Transliteration Shared Task (Li et al., 2009) is strictly orthographic. We view the conversion of a name in one language to its representation in another as the product of a series of single-character edits, and seek to learn a character edit model that maximizes the score of correct name pairs. Our approach follows that described in Freitag and Khadivi (2007), a "structured perceptron" with cheaply computed character n-grams as features. Here, we give a brief description, and present the successful enhancements we tried specifically for the shared task.

2.1 Perceptron Edit Model

Suppose we are given two sequences, $\boldsymbol{s}_1^m \in \Sigma_s^*$ and $\boldsymbol{t}_1^n \in \Sigma_t^*$. We desire a function $A(\boldsymbol{s}, \boldsymbol{t}) \mapsto \mathcal{N}$ which assigns high scores to correct pairs $\boldsymbol{s}, \boldsymbol{t}$. 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:

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

Proceedings of the 2009 Named Entities Workshop, ACL-IJCNLP 2009, pages 132–135,
Suntec, Singapore, 7 August 2009.

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.

In the experiments reported in this paper, we assume that each local function a is defined in terms of $p + q$ features, $\{f_1, \cdots, f_p, f_{p+1}, \cdots, f_{p+q}\}$, defined over the source and target alphabets, respectively, and that these features have the functional form $\Sigma^* \times \mathcal{N} \mapsto \mathcal{R}$.

In this paper we exclusively use character n-gram indicator features. The "order" of a model is the size of the largest n-grams; for a model of order 2, features would be the bigrams and unigrams immediately adjacent to a given string position. Since the shared task is to *generate* target strings, only features for preceding n-grams are used in the target language.

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.

We optimize these weights through an extension of perceptron training for sequence labeling, due to Collins (2002). 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 for a source string, target string, and corresponding edit sequence.

Table 1 shows the training algorithm. Starting with a zero parameter vector, we iterate through the collection of source sequences. For each sequence, we pick two target sequences, one the "true" transliteration t of the source string s, and one chosen by searching for a string t' that yields a maximal score according to the current model A_α (Line 6). If the model fails to assign t a higher score than t' (Line 9), we apply the perceptron training update (Line 10).

Note that because generation *constructs* the target sequence, the search in Line 6 for a target string t' that yields maximal $A_\alpha(\boldsymbol{s}, \boldsymbol{t'})$ is not trivial, and does not correspond to a simple recurrence relation like Equation 1. Both in training and testing, we use a beam search for target string generation. In training, this may mean that we find a t' with lower score than the correct target t. In such cases (Line 9 returns false), the model has correctly ordered the two alternative transliterations, and does not require updating.

```
1:  Given training set S = {⟨s, t⟩}
2:  V ← [], an empty list
3:  α ← 0, a weight vector
4:  for some number of iterations do
5:      for ⟨s, t⟩ in S do
6:          t' ← maxarg_{t'} A_α(s, t')
7:          ⟨e, v⟩ ← A_α(s, t)
8:          ⟨e', v'⟩ ← A_α(s, t')
9:          if v' ≥ v then
10:             α ← α + Φ(s, t, e) − Φ(s, t', e')
11:         end if
12:         Append α to V
13:     end for
14: end for
15: 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.

2.2 Shared Task Extensions

This approach has been used effectively for practical transliteration of names from English to Arabic and Mandarin (and vice versa). As part of the NEWS shared task, we experimented with two simple extensions, both of which yielded improvements over the baseline described above. These extensions were used in our official submission for alphabetic language pairs. We treated English-to-Chinese somewhat differently, as described below.

Simple character n-gram constraints. The described approach sometimes violates target language spelling conventions by interpolating clearly inappropriate characters into a string that is otherwise a reasonable transliteration. We take this behavior as symptomatic of a kind of undertraining in some portion of the problem space, a possible byproduct of 1-best perceptron training. One principled solution may be to optimize against n-best lists (Bellare et al., 2009).

Instead, we address this shortcoming in a straightforward way—by prohibiting the creation of n-grams, for some small n, that do not occur

in the training data. Under a bigram restriction, if 'ab' is not seen in training, then an operation that inserts 'b' after 'a' is disallowed. In essence, we impose a very simple character language model of the target domain.

Our non-standard English-to-Chinese contributions, which involved transliterating from English to pinyin, employed a similar idea. In these experiments, rather than character bigrams, the model was constrained to produce only legal pinyin sequences.

Bidirectional generation. Character n-gram restrictions yielded modest but universal improvements on development data. Larger improvements were obtained through an equally simple idea: Instead of a single left-to-right model, we trained two models, one generating left-to-right, the other right-to-left, each model constrained by n-gram tables, as described above. At evaluation time, each of the constituent models was used to generate a large number of candidate strings (100, typically). All strings in the union of these two sets were then assigned a score, which was the unweighted sum of scores according to the constituent models, and reranked according to this score. The 10 highest-scoring were retained for evaluation.

A buggy implementation of this two-model idea accounts for our poor showing in the official evaluation. Because of a trivial error in the script we used to produce output, right-to-left models were treated as if they were left-to-right. The resulting strings and scores were consequently erroneous.

3 Evaluation

We experimented with models of order 2 and 3 (2-gram and 3-gram features) on shared task data for English to Hindi, Kannada, Russian, and Tamil (Kumaran and Kellner, 2007). Based on development accuracy scores, we found models of order 3 to be consistently better than order 2, and our submitted results use only order-3 models, with one exception. English-to-native-Chinese (Li et al., 2004) was treated as a special case. Using trigram features in the target language results in an explosion in the feature space, and a model that is slow to train and performs poorly. Thus, only for this language pair, we devised a mixed-order model, one using trigram features over English strings, and unigram features over Chinese. Because of the large target-language branching factor, the mixed-order native Chinese model remaining one of the slowest to train.

Languages	Accuracy	Delta
EnHi	0.465	-0.033
EnHi baseline	0.421	-0.077
EnKa	0.396	-0.002
EnKa baseline	0.370	-0.028
EnRu	0.609	-0.004
EnRu baseline	0.588	-0.025
EnTa	0.475	+0.001
EnTa baseline	0.422	-0.052
EnCh standard	0.672	-0.059
EnCh non-standard 1	0.673	-0.236
EnCh non-standard 2	0.5	-0.409

Table 2: Post-contest accuracy on evaluation set, including delta from highest-scoring contest participant.

We trained all models for 20 epochs, evaluating the 1-best accuracy of intervening models on the development data. In all cases, we observed that accuracy increased steadily for some number of iterations, after which it plateaued. Consequently, for all language pairs, we submitted the predictions of the latest model.

Table 2 lists accuracy reported by the official evaluation script on the contest evaluation data. All non-Chinese runs in the table are "standard," and are trained exclusively on shared task training data. Those labeled "baseline" are left-to-right models with no character n-gram constraints. These results were obtained after release of the evaluation data, but differ from our official submission in only two ways: First, and most importantly, the bug described previously was corrected. Second, in some cases training runs that had not completed at evaluation time were allowed to run to the full 20 epochs, and the resulting models were used. The exceptions are Hindi and native Chinese, each of which reflect performance at approximately 10 epochs. Without exception, a beam of size 100 was used to generate these results.

With the exception of "EnCh standard," all results in the table employed the bidirectional scheme described above. The Chinese non-standard runs differ from standard only in that models were trained to perform English-to-pinyin transliteration, followed by a conversion from pinyin to native Chinese using tables provided by

the Unicode consortium. Non-standard Run 1 retains pinyin tonal diacritics, while Run 2 omits them. The mapping from pinyin to native Chinese characters introduces indeterminacy, which we accounted for in a simple fashion: First, in constructing a pinyin-to-Chinese conversion table, we discarded any Chinese characters that were used in the training data fewer than some small fraction of cases. Then, given a ranked list of pinyin transliterations, we generated all possible native Chinese sequences, ranked by the product of observed pinyin-to-Chinese probabilities, according to training frequencies.

It will be observed that our non-standard English-to-Chinese results lag considerably behind the best results. We suspect this is due in part to the fact that no additional training data was used in these experiments–only a change in representation.

4 Discussion and Conclusion

Treating the transliteration problem as one of orthographic transduction appears viable, particularly for conversion between alphabetic languages. An empirical character edit model based on a structured perceptron and character n-gram features, and using a simple training procedure that discovers appropriate weights for latent character alignment operations, yields performance that is generally as good as alternative approaches explored in the shared task. The key is model combination, particularly the combination of left-to-right and right-to-left models, respectively.

In contrast to the alphabetic language pairs, our performance on Chinese falls somewhat short. Nevertheless, it is possible that simple modifications of the basic procedure would render it competitive on English-Chinese, as well. In converting from English to native Chinese, we relied on a mixed-order model, with order-3 English features. It is possible that trigrams are too small in some cases to identify the appropriate Chinese character, and that 4-grams, if we can afford them, will make the difference. There is virtue in the idea of transliterating to pinyin as an intermediate step; converting to tonal pinyin yields accuracy at the same level as English-to-native-Chinese, even with the indeterminacy it introduces. Future work includes more principled approaches to resolving this indeterminacy, and combined pinyin/native models.

Acknowledgments

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023 (approved for public release, distribution unlimited). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of 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*.

K. Bellare, K. Crammer, and D. Freitag. 2009. Loss-sensitive discriminative training of machine transliteration models. In *Proceedings of the Student Research Workshop and Doctoral Consortium at NLT/NAACL 2009*.

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

D. Freitag and S. Khadivi. 2007. A sequence alignment model based on the averaged perceptron. In *Proceedings of EMNLP-CoNLL 2007*.

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

A. Kumaran and T. Kellner. 2007. A generic framework for machine transliteration. In *Proceedings of the 30th SIGIR*.

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*.

H. Li, M. Zhang, and J. Su. 2004. A joint source channel model for machine transliteration. In *Proceedings of the 42nd ACL*.

H. Li, A. Kumaran, M. Zhang, and V. Pervouchine. 2009. Whitepaper of NEWS 2009 Machine Transliteration Shared Task. In *Proceedings of ACL-IJCNLP 2009 Named Entities Workshop (NEWS 2009)*.