DIRECTL: a Language-Independent Approach to Transliteration

Sittichai Jiampojamarn, Aditya Bhargava, Qing Dou, Kenneth Dwyer, Grzegorz Kondrak
Department of Computing Science
University of Alberta
Edmonton, AB, T6G 2E8, Canada
{sj,abhargava,qdou,dwyer,kondrak}@cs.ualberta.ca

Abstract

We present DIRECTL: an online discriminative sequence prediction model that employs a many-to-many alignment between target and source. Our system incorporates input segmentation, target character prediction, and sequence modeling in a unified dynamic programming framework. Experimental results suggest that DIRECTL is able to independently discover many of the language-specific regularities in the training data.

1 Introduction

In the transliteration task, it seems intuitively important to take into consideration the specifics of the languages in question. Of particular importance is the relative character length of the source and target names, which vary widely depending on whether languages employ alphabetic, syllabic, or ideographic scripts. On the other hand, faced with the reality of thousands of potential language pairs that involve transliteration, the idea of a language-independent approach is highly attractive.

In this paper, we present DIRECTL: a transliteration system that, in principle, can be applied to any language pair. DIRECTL treats the transliteration task as a sequence prediction problem: given an input sequence of characters in the source language, it produces the most likely sequence of characters in the target language. In Section 2, we discuss the alignment of character substrings in the source and target languages. Our transcription model, described in Section 3, is based on an online discriminative training algorithm that makes it possible to efficiently learn the weights of a large number of features. In Section 4, we provide details of alternative approaches that incorporate language-specific information. Finally, in Section 5 and 6, we compare the experimental results of DIRECTL with its variants that incorporate language-specific pre-processing, phonetic alignment, and manual data correction.

2 Transliteration alignment

In the transliteration task, training data consist of word pairs that map source language words to words in the target language. The matching between character substrings in the source word and target word is not explicitly provided. These hidden relationships are generally known as *alignments*. In this section, we describe an EM-based many-to-many alignment algorithm employed by DIRECTL. In Section 4, we discuss an alternative phonetic alignment method.

We apply an unsupervised many-to-many alignment algorithm (Jiampojamarn et al., 2007) to the transliteration task. The algorithm follows the expectation maximization (EM) paradigm. In the expectation step shown in Algorithm 1, partial counts γ of the possible substring alignments are collected from each word pair (x^T, y^V) in the training data; T and V represent the lengths of words x and y, respectively. The forward probability α is estimated by summing the probabilities of all possible sequences of substring pairings from left to right. The FORWARD-M2M procedure is similar to lines 5 through 12 of Algorithm 1, except that it uses Equation 1 on line 8, Equation 2 on line 12, and initializes $\alpha_{0,0} := 1$. Likewise, the backward probability β is estimated by summing the probabilities from right to left.

$$\alpha_{t,v} \mathrel{+}= \delta(x^t_{t-i+1}, \epsilon)\alpha_{t-i,v} \tag{1}$$

$$\alpha_{t,v} \mathrel{+}= \delta(x^t_{t-i+1}, y^v_{v-j+1})\alpha_{t-i,v-j} \tag{2}$$

The $maxX$ and $maxY$ variables specify the maximum length of substrings that are permitted when creating alignments. Also, for flexibility, we allow a substring in the source word to be aligned with a “null” letter (ϵ) in the target word.

Algorithm 1: Expectation-M2M alignment

```
Input: $x^T, y^V, maxX, maxY, \gamma$
Output: $\gamma$
$\alpha :=$ FORWARD-M2M $(x^T, y^V, maxX, maxY)$
$\beta :=$ BACKWARD-M2M $(x^T, y^V, maxX, maxY)$
if $(\alpha_{T,V} = 0)$ then
    return
for $t = 0 \dots T$ , $v = 0 \dots V$ do
    if $(t > 0)$ then
        for $i = 1 \dots maxX$ st $t - i \geq 0$ do
            $\gamma(x_{t-i+1}^t, \epsilon) \mathrel{+}= \frac{\alpha_{t-i,v}\delta(x_{t-i+1}^t, \epsilon)\beta_{t,v}}{\alpha_{T,V}}$
    if $(v > 0 \wedge t > 0)$ then
        for $i = 1 \dots maxX$ st $t - i \geq 0$ do
            for $j = 1 \dots maxY$ st $v - j \geq 0$ do
    $\gamma(x_{t-i+1}^t, y_{v-j+1}^v) \mathrel{+}= \frac{\alpha_{t-i,v-j}\delta(x_{t-i+1}^t, y_{v-j+1}^v)\beta_{t,v}}{\alpha_{T,V}}$
```

Algorithm 2: Online discriminative training

```
Input: Data $\{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \dots, (\mathbf{x}_m, \mathbf{y}_m)\}$,
       number of iterations $k$, size of $n$-best list $n$
Output: Learned weights $\psi$
$\psi := \vec{0}$
for $k$ iterations do
    for $j = 1 \dots m$ do
        $\hat{Y}_j = \{\hat{\mathbf{y}}_{j1}, \dots, \hat{\mathbf{y}}_{jn}\} = \arg\max_{\mathbf{y}}[\psi \cdot \Phi(\mathbf{x}_j, \mathbf{y})]$
        update $\psi$ according to $\hat{Y}_j$ and $\mathbf{y}_j$
return $\psi$
```

In the maximization step, we normalize the partial counts γ to the alignment probability δ using the conditional probability distribution. The EM steps are repeated until the alignment probability δ converges. Finally, the most likely alignment for each word pair in the training data is computed with the standard Viterbi algorithm.

3 Discriminative training

We adapt the online discriminative training framework described in (Jiampojamarn et al., 2008) to the transliteration task. Once the training data has been aligned, we can hypothesize that the i^{th} letter substring $x_i \in \mathbf{x}$ in a source language word is transliterated into the i^{th} substring $y_i \in \mathbf{y}$ in the target language word. Each word pair is represented as a feature vector $\Phi(\mathbf{x}, \mathbf{y})$. Our feature vector consists of (1) n-gram context features, (2) HMM-like transition features, and (3) linear-chain features. The n-gram context features relate the letter evidence that surrounds each letter x_i to its output y_i. We include all n-grams that fit within a context window of size c. The c value is determined using a development set. The HMM-like transition features express the cohesion of the output $\mathbf{y}$ in the target language. We make a first order Markov assumption, so that these features are bigrams of the form (y_{i-1}, y_i). The linear-chain features are identical to the context features, except that y_i is replaced with a bi-gram (y_{i-1}, y_i).

Algorithm 2 trains a linear model in this feature space. The procedure makes k passes over the aligned training data. During each iteration, the model produces the n most likely output words $\hat{Y}_j$ in the target language for each input word $\mathbf{x}_j$ in the source language, based on the current parameters ψ. The values of k and n are determined using a development set. The model parameters are updated according to the correct output $\mathbf{y}_j$ and the predicted n-best outputs $\hat{Y}_j$, to make the model prefer the correct output over the incorrect ones. Specifically, the feature weight vector ψ is updated by using MIRA, the Margin Infused Relaxed Algorithm (Crammer and Singer, 2003). MIRA modifies the current weight vector ψ_o by finding the smallest changes such that the new weight vector ψ_n separates the correct and incorrect outputs by a margin of at least $\ell(\mathbf{y}, \hat{\mathbf{y}})$, the loss for a wrong prediction. We define this loss to be 0 if $\hat{\mathbf{y}} = \mathbf{y}$; otherwise it is $1 + d$, where d is the Levenshtein distance between $\mathbf{y}$ and $\hat{\mathbf{y}}$. The update operation is stated as a quadratic programming problem in Equation 3. We utilize a function from the SVM^{light} package (Joachims, 1999) to solve this optimization problem.

$$\begin{aligned} &\min_{\psi_n} \parallel \psi_n - \psi_o \parallel \\ &\text{subject to } \forall \hat{\mathbf{y}} \in \hat{Y} : \\ &\psi_n \cdot (\Phi(\mathbf{x}, \mathbf{y}) - \Phi(\mathbf{x}, \hat{\mathbf{y}})) \geq \ell(\mathbf{y}, \hat{\mathbf{y}}) \end{aligned} \tag{3}$$

The $\arg\max$ operation is performed by an exact search algorithm based on a phrasal decoder (Zens and Ney, 2004). This decoder simultaneously finds the l most likely substrings of letters $\mathbf{x}$ that generate the most probable output $\mathbf{y}$, given the feature weight vector ψ and the input word x^T. The search algorithm is based on the following dynamic programming recurrence:

$$Q(0, \$) = 0$$

$$Q(t, p) = \max_{\substack{p', p, \\ t - maxX \leq t' < t}} \{\psi \cdot \phi(x_{t'+1}^t, p', p) + Q(t', p')\}$$

$$Q(T{+}1, \$) = \max_{p'} \{\psi \cdot \phi(\$, p', \$) + Q(T, p')\}$$

To find the n-best predicted outputs, the table Q records the top n scores for each output substring that has the suffix p substring and is generated by the input letter substring x_1^t; here, p' is a sub-output generated during the previous step. The notation $\phi(x_{t'+1}^{t}, p', p)$ is a convenient way to describe the components of our feature vector $\Phi(\mathbf{x}, \mathbf{y})$. The n-best predicted outputs $\hat{Y}$ can be discovered by backtracking from the end of the table, which is denoted by $Q(T+1, \$)$.

4 Beyond DIRECTL

4.1 Intermediate phonetic representation

We experimented with converting the original Chinese characters to Pinyin as an intermediate representation. Pinyin is the most commonly known Romanization system for Standard Mandarin. Its alphabet contains the same 26 letters as English. Each Chinese character can be transcribed phonetically into Pinyin. Many resources for Pinyin conversion are available online.[1] A small percentage of Chinese characters have multiple pronunciations represented by different Pinyin representations. For those characters (about 30 characters in the transliteration data), we manually selected the pronunciations that are normally used for names. This preprocessing step significantly reduces the size of target symbols from 370 distinct Chinese characters to 26 Pinyin symbols which enables our system to produce better alignments.

In order to verify whether the addition of language-specific knowledge can improve the overall accuracy, we also designed intermediate representations for Russian and Japanese. We focused on symbols that modify the neighboring characters without producing phonetic output themselves: the two *yer* characters in Russian, and the long vowel and *sokuon* signs in Japanese. Those were combined with the neighboring characters, creating new "super-characters."

4.2 Phonetic alignment with ALINE

ALINE (Kondrak, 2000) is an algorithm that performs phonetically-informed alignment of two strings of phonemes. Since our task requires the alignment of characters representing different writing scripts, we need to first replace every character with a phoneme that is the most likely to be produced by that character.

We applied slightly different methods to the test languages. In converting the Cyrillic script into phonemes, we take advantage of the fact that the Russian orthography is largely phonemic, which makes it a relatively straightforward task. In Japanese, we replace each Katakana character with one or two phonemes using standard transcription tables. For the Latin script, we simply treat every letter as an IPA symbol (International Phonetic Association, 1999). The IPA contains a subset of 26 letter symbols that tend to correspond to the usual phonetic value that the letter represents in the Latin script. The Chinese characters are first converted to Pinyin, which is then handled in the same way as the Latin script.

Similar solutions could be engineered for other scripts. We observed that the transcriptions do not need to be very precise in order for ALINE to produce high quality alignments.

4.3 System combination

The combination of predictions produced by systems based on different principles may lead to improved prediction accuracy. We adopt the following combination algorithm. First, we rank the individual systems according to their top-1 accuracy on the development set. To obtain the top-1 prediction for each input word, we use simple voting, with ties broken according to the ranking of the systems. We generalize this approach to handle n-best lists by first ordering the candidate transliterations according to the highest rank assigned by any of the systems, and then similarly breaking ties by voting and system ranking.

5 Evaluation

In the context of the NEWS 2009 Machine Transliteration Shared Task (Li et al., 2009), we tested our system on six data sets: from English to Chinese (EnCh) (Li et al., 2004), Hindi (EnHi), Russian (EnRu) (Kumaran and Kellner, 2007), Japanese Katakana (EnJa), and Korean Hangul (EnKo); and from Japanese Name to Japanese Kanji (JnJk)[2]. We optimized the models' parameters by training on the training portion of the provided data and measuring performance on the development portion. For the final testing, we trained the models on all the available labeled data (training plus development data). For each data set, we converted any uppercase letters to lowercase. Our system outputs the top 10 candidate answers for each input word.

Table 1 reports the performance of our system on the development and final test sets, measured in terms of top-1 word accuracy (ACC). For certain language pairs, we tested variants of the base

[1]For example, http://www.chinesetopinyin.com/

[2]http://www.cjk.org/

Task	Model	Dev	Test
EnCh	DirecTL	72.4	71.7
	Int(m2m)	73.9	73.4
	Int(Aline)	73.8	73.2
	Combined	74.8	74.6
EnHi	DirecTL	41.4	49.8
	DirecTL+mc	42.3	50.9
EnJa	DirecTL	49.9	50.0
	Int(m2m)*	49.6	49.2
	Int(Aline)	48.3	51.0
	Combined*	50.6	50.5
EnKo	DirecTL	36.7	38.7
EnRu	DirecTL	80.2	61.3
	Int(m2m)	80.3	60.8
	Int(Aline)	80.0	60.7
	Combined*	80.3	60.8
JnJk	DirecTL	53.5	56.0

Table 1: Top-1 word accuracy on the development and test sets. The asterisk denotes the results obtained after the test reference sets were released.

system described in Section 4. DirecTL refers to our language-independent model, which uses many-to-many alignments. The Int abbreviation denotes the models operating on the language-specific intermediate representations described in Section 4.1. The alignment algorithm (Aline or m2m) is given in brackets.

In the EnHi set, many names consisted of multiple words: we assumed a one-to-one correspondence between consecutive English words and consecutive Hindi words. In Table 1, the results in the first row (DirecTL) were obtained with an automatic cleanup script that replaced hyphens with spaces, deleted the remaining punctuation and numerical symbols, and removed 43 transliteration pairs with a disagreement between the number of source and target words. The results in the second row (DirecTL+mc) were obtained when the cases with a disagreement were individually examined and corrected by a Hindi speaker.

We did not incorporate any external resources into the models presented in Table 1. In order to emphasize the performance of our language-independent approach, we consistently used the DirecTL model for generating our "standard" runs on all six language pairs, regardless of its relative performance on the development sets.

6 Discussion

DirecTL, our language-independent approach to transliteration achieves excellent results, especially on the EnCh, EnRu, and EnHi data sets, which represent a wide range of language pairs and writing scripts. Both the many-to-many and phonetic alignment algorithms produce high-quality alignments. The former can be applied directly to the training data without the need for an intermediate representation, while the latter does not require any training. Surprisingly, incorporation of language-specific intermediate representations does not consistently improve the performance of our system, which indicates that DirecTL may be able to discover the structures implicit in the training data without additional guidance. The EnHi results suggest that manual cleaning of noisy data can yield noticeable gains in accuracy. On the other hand, a simple method of combining predictions from different systems produced clear improvement on the EnCh set, but mixed results on two other sets. More research on this issue is warranted.

Acknowledgments

This research was supported by the Alberta Ingenuity, Informatics Circle of Research Excellence (iCORE), and Natural Sciences of Engineering Research Council of Canada (NSERC).

References

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991.

International Phonetic Association. 1999. *Handbook of the International Phonetic Association*. Cambridge University Press.

Sittichai Jiampojamarn, Grzegorz Kondrak, and Tarek Sherif. 2007. Applying many-to-many alignments and Hidden Markov Models to letter-to-phoneme conversion. In *Proc. HLT-NAACL*, pages 372–379.

Sittichai Jiampojamarn, Colin Cherry, and Grzegorz Kondrak. 2008. Joint processing and discriminative training for letter-to-phoneme conversion. In *Proc. ACL*, pages 905–913.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. Advances in kernel methods: support vector learning, pages 169–184. MIT Press.

Grzegorz Kondrak. 2000. A new algorithm for the alignment of phonetic sequences. In *Proc. NAACL*, pages 288–295.

A. Kumaran and Tobias Kellner. 2007. A generic framework for machine transliteration. In *Proc. SIGIR*, pages 721–722.

Haizhou Li, Min Zhang, and Jian Su. 2004. A joint source channel model for machine transliteration. In *Proc. ACL*, pages 159–166.

Haizhou Li, A Kumaran, Min Zhang, and Vladimir Pervouchine. 2009. Whitepaper of NEWS 2009 machine transliteration shared task. In *Proc. ACL-IJCNLP Named Entities Workshop.*

Richard Zens and Hermann Ney. 2004. Improvements in phrase-based statistical machine translation. In *Proc. HLT-NAACL*, pages 257–264.