# NEWS 2009 Machine Transliteration Shared Task System Description: Transliteration with Letter-to-Phoneme Technology

**Colin Cherry** and **Hisami Suzuki**
Microsoft Research
One Microsoft Way
Redmond, WA, 98052
{colinc,hisamis}@microsoft.com

## Abstract

We interpret the problem of transliterating English named entities into Hindi or Japanese Katakana as a variant of the letter-to-phoneme (L2P) subtask of text-to-speech processing. Therefore, we apply a re-implementation of a state-of-the-art, discriminative L2P system (Jiampojamarn et al., 2008) to the problem, without further modification. In doing so, we hope to provide a baseline for the NEWS 2009 Machine Transliteration Shared Task (Li et al., 2009), indicating how much can be achieved without transliteration-specific technology. This paper briefly summarizes the original work and our re-implementation. We also describe a bug in our submitted implementation, and provide updated results on the development and test sets.

## 1 Introduction

Transliteration occurs when a word is borrowed into a language with a different character set from its language of origin. The word is transcribed into the new character set in a manner that maintains phonetic correspondence.

When attempting to automate machine transliteration, modeling the channel that transforms source language characters into transliterated target language characters is a key component to good performance. Since the primary signal followed by human transliterators is phonetic correspondence, it makes sense that a letter-to-phoneme (L2P) transcription engine would perform well at this task. Of course, transliteration is often framed within the larger problems of translation and bilingual named entity co-reference, making available a number of other interesting features, such as target lexicons (Knight and Graehl, 1998), distributional similarity (Bilac and Tanaka, 2005), or the dates of an entity's mentions in the news (Klementiev and Roth, 2006). However, this task's focus on generation has isolated the character-level component, which makes L2P technology a near-ideal match. For our submission, we re-implement the L2P approach described by Jiampojamarn et al. (2008) as faithfully as possible, and apply it unmodified to the transliteration shared task for the English-to-Hindi (Kumaran and Kellner, 2007) and English-to-Japanese Katakana[1] tests.

## 2 Approach

### 2.1 Summary of L2P approach

The core of the L2P transduction engine is the dynamic programming algorithm for monotone phrasal decoding (Zens and Ney, 2004). The main feature of this algorithm is its capability to transduce many consecutive characters with a single operation. This algorithm is used to conduct a search for a max-weight derivation according to a linear model with indicator features. A sample derivation is shown in Figure 1.

There are two main categories of features: context and transition features, which follow the first two feature templates described by Jiampojamarn et al. (2008). Context features are centered around a transduction operation. These features include an indicator for the operation itself, which is then conjoined with indicators for all $n$-grams of source context within a fixed window of the operation. Transition features are Markov or $n$-gram features. They ensure that the produced target string makes sense as a character sequence, and are represented as indicators on the presence of target $n$-grams. The feature templates have two main parameters, the size $S$ of the character window from which source context features are drawn, and the maximum length $T$ of target $n$-gram indicators. We fit these parameters using grid search over 1-best

[1] Provided by http://www.cjk.org



*Proceedings of the 2009 Named Entities Workshop, ACL-IJCNLP 2009*, pages 69–71,
Suntec, Singapore, 7 August 2009. 

ame → アメ , ri → リ , can → カン

Figure 1: Example derivation transforming "American" into "アメリカン".

accuracy on the provided development sets.

The engine's features are trained using the structured perceptron (Collins, 2002). Jiampojamarn et al. (2008) show strong improvements in the L2P domain using MIRA in place of the perceptron update; unfortunately, we did not implement a $k$-best MIRA update due to time constraints. In our implementation, no special consideration was given to the availability of multiple correct answers in the training data; we always pick the first reference transliteration and treat it as the only correct answer. Investigating the use of all correct answers would be an obvious next step to improve the system.

### 2.2 Major differences in implementation

Our system made two alternate design decisions (we do not claim improvements) over those made by (Jiampojamarn et al., 2008), mostly based on the availability of software. First, we employed a beam of 40 candidates in our decoder, to enable efficient use of large language model contexts. This is put to good use in the Hindi task, where we found $n$-gram indicators of length up to $n = 6$ provided optimal development performance.

Second, we employed an alternate character aligner to create our training derivations. This aligner is similar to recent non-compositional phrasal word-alignment models (Zhang et al., 2008), limited so it can only produce monotone character alignments. The aligner creates substring alignments, without insertion or deletion operators. As such, an aligned transliteration pair also serves as a transliteration derivation. We employed a maximum substring length of 3.

The training data was heuristically cleaned after alignment. Any derivation found by the aligner that uses an operation occurring fewer than 3 times throughout the entire training set was eliminated. This reduced training set sizes to 8,511 pairs for English-Hindi and 20,306 pairs for English-Katakana.

Table 1: Development and test 1-best accuracies, as reported by the official evaluation tool

| System / Test set | With Bug | Fixed |
|---|---|---|
| Hindi Dev | 36.7 | 39.6 |
| Hindi Test | 41.8 | 46.6 |
| Katakana Dev | 46.0 | 47.1 |
| Katakana Test | 46.6 | 46.9 |

## 3 The Bug

The submitted version of our system had a bug in its transition features: instead of generating an indicator for every possible $n$-gram in the generated target sequence, it generated $n$-grams over target substrings, defined by the operations used during transduction. Consider, for example, the derivation shown in Figure 1, which generates "アメリカン". With buggy trigram transition features, the final operation would produce the single indicator [メリ|カン], instead of the two character-level trigrams [メリ|カ] and [リカ|ン]. This leads to problems with data sparsity, which we had not noticed on unrelated experiments with larger training data. We report results both with the **bug** and with **fixed** transition features. We do so to emphasize the importance of a fine-grained language discriminative language model, as opposed to one which operates on a substring level.

## 4 Development

Development consisted of performing a parameter grid search over $S$ and $T$ for each language pair's development set. All combinations of $S = 0 \ldots 4$ and $T = 0 \ldots 7$ were tested for each language pair. Based on these experiments, we selected (for the fixed version), values of $S = 2$, $T = 6$ for English-Hindi, and $S = 4$, $T = 3$ for English-Katakana.

## 5 Results

The results of our internal experiments with the official evaluation tool are shown in Table 1. We report 1-best accuracy on both development and test sets, with both the buggy and fixed versions of our system. As one can see, the bug makes less of an impact in the English-Katakana setting, where more training data is available.

## 6 Conclusion

We have demonstrated that an automatic letter-to-phoneme transducer performs fairly well on this transliteration shared task, with no language-specific or transliteration-specific modifications. Instead, we simply considered Hindi or Katakana to be an alternate encoding for English phonemes. In the future, we would like to investigate proper use of multiple reference answers during perceptron training.

## Acknowledgments

We would like to thank the NEWS 2009 Machine Transliteration Shared Task organizers for creating this venue for comparing transliteration methods. We would also like to thank Chris Quirk for providing us with his alignment software.

## References

Slaven Bilac and Hozumi Tanaka. 2005. Extracting transliteration pairs from comparable corpora. In *Proceedings of the Annual Meeting of the Natural Language Processing Society*, Japan.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *EMNLP*.

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

Alexandre Klementiev and Dan Roth. 2006. Named entity transliteration and discovery from multilingual comparable corpora. In *HLT-NAACL*, pages 82–88, New York City, USA, June.

Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4):599–612.

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

Haizhou Li, A. Kumaran, Vladimir Pervouchine, and Min Zhang. 2009. Report on NEWS 2009 machine transliteration shared task. In *Proceedings of ACL-IJCNLP 2009 Named Entities Workshop (NEWS 2009)*, Singapore.

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

Hao Zhang, Chris Quirk, Robert C. Moore, and Daniel Gildea. 2008. Bayesian learning of non-compositional phrases with synchronous parsing. In *ACL*, pages 97–105, Columbus, Ohio, June.