Transliteration Generation from the Orthographic and Phonetic Data

by

Lei Yao

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

Machine transliteration is important to machine translation and cross-lingual information retrieval. Previous works show that machine transliteration can benefit from supplemental phonetic transcriptions and transliterations from other languages through a re-ranking framework. In this thesis, I propose to leverage supplemental information by jointly considering it with the spelling of the source word during the transliteration generation process. This approach is shown to be more accurate and faster than the re-ranking approach. In addition, I propose to represent supplemental transliterations through a common phonetic interlingua. Experiments suggest that the interlingua representations can be as effective as the original orthography, and can even be obtained from languages that are not available during training.

To my friends and family

Acknowledgements

First of all, I am very grateful to my supervisor, Dr. Kondrak, for his great guidance and patience throughout my graduate study and research. I would like extend my thanks to Garrett Nicolai, for his collaboration and constructive feedbacks; thanks to Mohammad Salameh, for the fruitful discussions. And, especial thanks to my family for being so supportive all the time.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Transliteration, also known as phonetic translation, is the process of transforming the script of a word from the source language to the target language, while preserving the original pronunciation as closely as possible. For the example in Figure 1.1, the name *Joaquin* in English is transliterated into ホアキン (pronounced as HO A KI N) in Japanese. Machine transliteration aims to automate this process.

Machine transliteration is important to statistical machine translation (SMT). SMT usually performs poorly at translating names (Hermjakob et al., 2008) while some names often carry the central meanings of a sentence (Li et al., 2013). Therefore, machine transliteration is applied to overcome this bottleneck. It is also important for cross-lingual information retrieval (Abduljaleel and Larkey, 2003; Udupa and Khapra, 2010). For example, we may not have documents for the query ホアキン, but we can return documents containing the name *Joaquin* with the help of machine transliteration.

Most early transliteration models are developed on a phonetic-based framework. Their intuition is straightforward, i.e., the phonetic space is common among languages, which can be used to generate intermediate representation between source and target languages. The goal of machine transliteration is then simplified to learn transformation rules between the source language and the target language in the common phonetic space. The early models often apply a pipeline architecture (grapheme to phoneme conversion, followed by phoneme to phoneme conversion, then phoneme to grapheme conversion) that increases the chance of error propagation. Also, they do not account for the fact that some names are transliterated according to their orthographic rather than their phonetic representation. Spelling-based models are thereby developed to overcome the above-mentioned drawbacks. These models learn directly the orthographic mapping between the source language and the target language. Thus, there is only one step in their process and orthography is taken into consideration.

Although spelling-based models generally achieve better results than phonetic-based ones, they can barely handle irregular names which are pronounced unusual. Let us take a look at the previous example of transliterating *Joaquin* into Japanese. The source word, *Joaquin*, originates from Spanish thus it does not follow English

Figure 1.1: The source name *Joaquin* and the corresponding target transliteration ホアキン (pronounced as HO A KI N). It is difficult to pronounce the source name, but with the help of its phonemes */hwakin/* and supplemental transliterations from other languages we have a better idea how to transliterate it into Japanese.

orthography nor phonology. It is likely to be incorrectly transliterated into ジョアキン*(pronounced as ZI yo A KI N)*, because this word, especially the letter *J*, is pronounced unusual. If we have access to the phonetic transcription of *Joaquin*, such as the phoneme string */hwakin/*, generating the correct target transliteration becomes a much easier task. Bhargava and Kondrak (2012) propose using the phonetic transcription of the source word to re-rank the outputs of spelling-based models, which are typically in the form of ranked lists of candidate transliterations. Although their method boosts the transliteration performance by a large margin, the post-hoc property of the re-ranking idea prevents further improvement. If the spelling-based models fail to put the correct transliteration in the candidate lists in the first place, re-ranking will not bring any improvement. Therefore, in this work, I propose to directly include phonemes into the process of generating transliterations. This joint consideration of orthographic and phonetic representations is more intuitive and natural. Imagine a translator attempts to transliterate an English name into Japanese. It is more likely that he/she will consider pronunciation and spelling simultaneously and then come up with an answer, rather than first creating a list of candidates based on the spelling and then picking the one with the closest sound to the original name. My experiments suggest that the joint approach is more accurate and faster than the re-ranking approach. In addition, I show that these two approaches can be combined to achieve further improvement.

Supplemental transliterations from other languages could also help improve transliteration. For the example in Figure 1.1, we are provided a Chinese transliteration 华坚. As a Chinese speaker, I know immediately the letter *J* in the source word *Joaquin* should be pronounced as */h/*, because 华 is a strong indication of the sound */h/*. Bhargava and Kondrak (2012) report significant improvement of using these supplemental transliterations to re-rank the results of spelling-based models. However, besides the above-mentioned drawbacks of their approach, it has additional limitations. First, the feature space expands polynomially when more languages are included, because features are extracted from the orthography of language pairs. Second, their model cannot utilize supplemental transliterations from languages that are not available during training, which means they need to retrain their model each time they have access to a new language. Inspired by the interlin-

gua machine translation and phonetic-based transliteration approaches, I propose to represent all the supplemental transliterations through a common phonetic interlingua, and then extract features from this interlingua representation. Thus, the size of the feature space is independent of the number of supplemental languages; and when we have access to new languages, we only need to learn how to map them to the phonetic interlingua rather than retraining the transliteration model. My experiments show the phonetic interlingua representation has great potential in encoding supplemental transliterations and leveraging transliterations from new languages.

The thesis is organized as follows. Chapter 2 introduces the related work. Chapter 3 describes the state-of-the-art string transduction model, DirecTL+, through which I implement my ideas. Chapter 4 details improving transliteration by jointly considering orthography with supplemental information such as phonetic transcriptions, and transliterations from other languages. I explore the idea of using interlingua to represent supplemental transliterations in Chapter 5. Chapter 6 presents my conclusion and future work.

Chapter 2

Related Work

This chapter starts with a brief review of the general machine transliteration. Then I will describe specifically the ideas of leveraging supplemental information and interlingua for machine transliteration.

2.1 Machine Transliteration

Works in machine transliteration fall into two groups, i.e., transliteration generation and transliteration mining. Transliteration generation models are developed to transliterate new terms. Transliteration mining aims to discover transliterations from large multilingual corpora; thus, it can be applied to provide training data for transliteration generation models. Since transliteration generation is often referred to as the core task of machine transliteration and is also the focus of my study, I only review transliteration generation in this thesis. I will use machine transliteration and transliteration generation interchangeably unless otherwise specified.

2.1.1 Transliteration Process

The transliteration generation process consists of two phases: training and decoding. The training stage assumes a bilingual corpus for training, where each source word is paired with its transliteration in the target language[1]. The decoding stage produces a ranked list of transliteration candidates given a source word. Figure 2.1 shows the typical transliteration generation framework in the literature (Karimi et al., 2011).

The training stage can be further divided into two phases: letter-level alignment between the source and target words, followed by transformation rule generation. The transformation rule is often represented as a mapping between a source letter/sound sequence and a target letter/sound sequence. Each transformation rule is associated with a score indicating the confidence of this transformation. The typical decoding stage also consists of two sub-stages: segmentation of the source

[1] I also refer to this kind of bilingual corpus as the *pairwise data*.

word into multiple letter sequences and generation of a list of candidate target words sorted by their confidence scores.

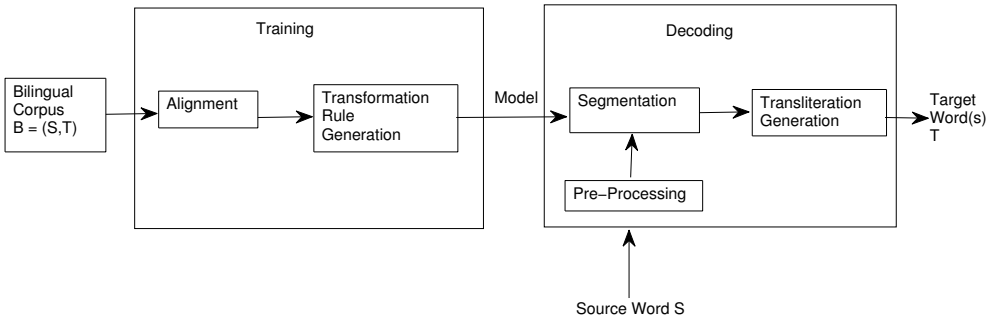

Figure 2.1: The typical machine transliteration framework summarized in (Karimi et al., 2011)

2.1.2 Alignment

The purpose of aligning the source and target words is to find the letter/sound correspondence between the source and target languages, which thereby provides the basis for learning transformation rules. The quality of alignment has a substantial effect on a model's performance. Figure 2.2 shows an aligned pair of English and Japanese transliterations. We can notice that there is no crossed lines in this example. This is due to the monotonic constraints of transliteration alignment (that is, the order of letter/sound is preserved between languages). Early transliteration systems directly apply the non-monotonic aligners that are originally developed for machine translation, such as IBM Model 1 and Model 3 (Brown et al., 1993) and GIZA++ (Och and Ney, 2003). However, non-monotonic alignments are inferior to monotonic alignments (Gao et al., 2004; Li et al., 2004; Karimi et al., 2007).

Transliteration alignment algorithms can be further categorized according to the representation they use.

- Phonetic-based: Algorithms in this category require preprocessing the source and target words by mapping them into a common phonetic space. Prior

Figure 2.2: The alignment between the English word *ADEPT* and the Japanese Katakana ア*(A)*ド*(DO)*ネ*(NE)*. _ is a null letter, which means the English letter *T* has no correspondence in Japanese.

knowledge about phonetic similarity is often utilized to achieve alignment with phonetic representations (Oh and Choi, 2002; Kessler, 2005; Tao et al., 2006; Yoon et al., 2007; Pervouchine et al., 2009; Jiampojamarn et al., 2009).

- Spelling-based: Since the orthography of the source language and the target languages is usually heterogeneous; in order to be language-independent, algorithms of this category assume no or minimal knowledge about the orthography. Unsupervised algorithms, e.g., Expectation Maximization(EM) (Dempster et al., 1977), are broadly applied to learn letter correspondences between languages (Abduljaleel and Larkey, 2003; Karimi et al., 2006; Jiampojamarn et al., 2009).

Notice that although alignment is important to transliteration, it is not a prerequisite to build a transliteration system. There are alignment free approaches that are also effective (Zelenko and Aone, 2006; Langlais, 2013).

2.1.3 Phonetic-based Transliteration

Most early efforts in machine transliteration follow a phonetic-based framework, due to the phonetic nature of transliteration. Two typical diagrams of these approaches are shown in Figure 2.3.

Methods represented by Figure 2.3(A) only create a single phonetic representation of either the source or the target language. Sound changes are not modelled explicitly in the phonetic space. Arbabi et al. (1994) propose a knowledge-based system for Arabic-to-English transliteration. Arabic names are first converted into phonetic transcriptions, which are then directly converted into English names. The

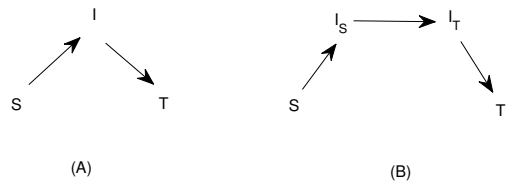

Figure 2.3: Two typical graphic representations of phonetic-based approaches. I is the phonetic representation.

two phases of conversion are based on a predefined set of rules. Thus, no machine learning technique is involved. Jeong et al. (1999) apply a hidden Markov model (HMM) for Korean-to-English transliteration. They only consider the phonetic representation of English, and assume any Korean letter is only dependent on one single pronunciation unit in English. Jung et al. (2000) approach English-to-Korean transliteration using an extended Markov model that allows the context information to be encoded; that is, the target letter depends on a context window of the source phonetic units.

Methods represented by Figure 2.3(B) create phonetic representations for both the source and target words. The transformation rules are then generated to transform sounds from the source language to the target language. Knight and Graehl (1998) create such a transliteration model for Japanese-to-English transliteration. Their model is composed of four successive components: one weighted finite-state transducer (WFST) to produce the phonetic transcription of a given Japanese word, one WFST to convert the Japanese transcription into the corresponding English transcription, one WFST to recover the English word from the transcription generated by the previous WFST, and one weighted finite-state acceptor (WFSA) to give the likelihood of an English word just as a unigram language model does. In order to train the second WFST, they apply the EM algorithm to align the source and target transcriptions. Note their EM algorithm only learns the co-occurrence of the source and target phonemes, thus phonetic similarities of phonemes are not explicitly modelled. Lin and Chen (2002) present a method of English-to-Chinese transliteration by explicitly utilizing a phonetic similarity measure. They apply a modified Widrow-Hoff learning algorithm to automatically acquire phonetic similarities from a bilingual transliteration corpus. Their automatically extracted phonetic similarities outperform hand-crafted phonetic similarities.

2.1.4 Spelling-based Transliteration

The phonetic approaches suffer from the error propagation problem because of their requirement of multiple processing steps. Spelling-based approaches are thereby proposed to overcome this problem. Figure 2.4 shows a general diagram of a spelling-based approach. We can see the number of steps in the transliteration process is reduced to one.

Figure 2.4: The graphic representation of spelling-based methods.

Kang and Choi (2000) study English-to-Korean transliteration with decision-tree learning. They derive their alignment algorithm by extending the cognate alignment algorithm by Covington (1996) with heuristic rules about correspondence between Korean and English letters, then apply decision tree learning to transform each English letter to Korean. Abduljaleel and Larkey (2003) apply $GIZA++$ to align English and Arabic words. They then adapt the noisy-channel model to learn English-to-Arabic transliteration based on the aligned data. Li et al. (2004) propose a joint source-channel model for English-to-Chinese transliteration. Their model is capable of learning alignment and transliteration simultaneously (the alignment model and the transliteration model actually share the same set of parameters), which has advantages over the systems that separate the two steps.

Sherif and Kondrak (2007) investigate Arabic-to-English transliteration using a Viterbi substring decoder and a substring-based transducer. They show that the transducer outperforms the Viterbi decoder as a result of its capability in implementing a unigram model and eliminating low probability mappings. Jiampojamarn et al. (2009) apply a general string aligner, M2Maligner (Jiampojamarn et al., 2007) and a general discriminative transducer, DirecTL+ (Jiampojamarn et al., 2008) for machine transliteration. They achieve the best results in several language pairs in the NEWS 2010 transliteration generation shared task (Li et al., 2010). They further improve their model by introducing joint n-gram features (Jiampojamarn et al., 2010). Since my proposed model is based on theirs, more details of their model will be provided in Chapter 3.

2.2 Leveraging supplemental languages

The idea of taking advantage of data from additional languages for machine transliteration is first explored by Khapra et al. (2010). They are motivated by the need to transliterate low-resource language pairs. They propose a bridge approach to transliterating low-resource language pair (X, Y) by pivoting on an additional language Z, where the pairwise data between (X, Z) and (Y, Z) is relatively large.

However, they show pivoting on Z results in less accuracy than directly transliterating X into Y. Zhang et al. (2010) and Kumaran et al. (2010) combine the pivot approach with the direct transliteration approach and achieve better performance than using either of the two approaches alone.

Bhargava and Kondrak (2011) find transliterations from other languages help Grapheme-to-Phoneme systems produce better phonetic transcriptions of names through a re-ranking framework. Their intuition is that transliterations from other languages alleviate the ambiguities in pronouncing names. They then generalize this idea to improve transliteration by utilizing supplemental transliterations from other languages and the phonetic transcriptions of the source language (Bhargava and Kondrak, 2012). Specifically, they apply a Support Vector Machine(SVM) re-ranker to re-rank outputs of a base spelling-based model, which are in the form of n-best lists of candidates transliterations. They report large-margin transliteration improvement on words with supplemental transliterations available over the base models.

2.3 Transliteration Interlingua

Although an obvious interlingua in transliteration exists, i.e., the phonetic space, the idea of transliteration interlingua is not well studied in terms of multilinguality. Most previous phonetic-based works study the intermediate representation between a single pair of languages but do not to generalize the representation for more than two languages. Also, few works pursue how transliterations of different language pairs can affect each other through the transliteration interlingua.

Yoon et al. (2007) propose representing each phoneme by a set of general phonetic features together with a set of carefully designed pseudo features. They use this representation as the phonetic transcription for five languages, e.g., English, Arabic, Chinese, Hindi and Korean. Then they apply this representation for transliteration mining. They report impressive results in mining transliterations of language pairs, for which training data is not available. For example, the model trained on English-Chinese pairs and the model trained on English-Arabic pairs can achieve close performance on the English-Arabic mining task. Note that the pseudo features are based on pronunciation errors of English-learners; thus, their work would not apply for language pairs without English involved.

Udupa and Khapra (2010) apply Canonical Correlation Analysis (CCA) (Harold, 1936) to find an interlingua representation for transliteration mining. They represent each source/target word in a feature vector based on character n-grams. CCA is then applied to map the source vector and target vector to a common space where they are maximally related. This common space is referred to as the interlingua space. And the mapped vectors in the interlingua space can be considered as the interlingua representation of the original words. Notice their interlingua representation is not phonetic and is derived automatically from the orthography of the source and target languages. Khapra et al. (2011) further explore this idea under a multi-

lingual setting, where a bridge language is introduced to alleviate the lack of data between the source and target language. CCA is applied to learn the interlingua representation of these three languages.

Inspired by the CCA idea, Jagarlamudi and Daume III (2012) further eliminate the need for pairwise transliteration data for transliteration mining. Their model only requires transcription data for training. Similar to the CCA approach, the interlingua space in their work is also implicit and uninterpretable to human. They report improvement of transliteration mining by incorporating transcription data from other languages.

We can see the major benefit brought by the transliteration interlingua is its multilinguality. Simply put, it provides an elegant way for transliteration systems to utilize resources from other languages. However, no previous work has applied the interlingua idea in the task of transliteration generation, which is a focus of this work.

Chapter 3

DirecTL+

DirecTL+ is a general discriminative string transducer developed by Jiampojamarn et al. (2010). By using it together with an unsupervised string aligner, M2M-aligner (Jiampojamarn et al., 2007), they report the state-of-the-art performance in the Grapheme-to-Phoneme (G2P) conversion task and the transliteration task of several language pairs (Jiampojamarn et al., 2010). Since my entire work is built upon DirecTL+, a detailed description of this model is provided in this chapter.

DirecTL+ requires its training data to be aligned pairs of words, as the one in Figure 2.2. Thus M2M-aligner is used to preprocess unaligned data. I use M2M-aligner as black box in this work. It has the following two properties: 1.it is able to learn alignment out of unaligned data. 2.it allows many-to-many alignment, such as the two letters *N,E* is aligned to ネ in Figure 2.2.

DirecTL+ is composed of three components.

- A scoring model to score an aligned pair of source and target strings (S, T).
- A search algorithm for finding the highest scoring target string given a source string.
- A training process to learn the weight parameters for scoring.

3.1 Scoring Model

The scoring model is represented as a linear combination of features $\alpha \cdot \Phi(S, T)$, where $\Phi(S, T)$ is the feature vector and α is a weighting vector. Assume both the input and output consist of m aligned substrings, such that S_i generates T_i. The scoring model extract features for each aligned substrings, and applies first-order Markov assumption that T_i only depends on T_{i-1}. The scoring model can be then rewritten as follows:

$$\sum_{i}^{m} \alpha \cdot \Phi(i, S, T_i, T_{i-1})$$

context	$s_{i-c}t_i$	linear-chain	$s_{i-c}t_{i-1}t_i$
	...		...
	$s_{i+c}t_i$		$s_{i+c}t_{i-1}t_i$
	$s_{i-c}s_{i-c+1}t_i$		$s_{i-c}s_{i-c+1}t_{i-1}t_i$
	...		...
	$s_{i+c-1}s_{i+c}t_i$		$s_{i+c-1}s_{i+c}t_{i-1}t_i$
	...		...
	$s_{i-c}...s_{i+c}t_i$		$s_{i-c}...s_{i+c}t_{i-1}t_i$
transition	$t_{i-1}t_i$		
joint n-gram	$s_{i+1-n}t_{i+1-n}s_it_i$		
	...		
	$s_{i-1}t_{i-1}s_it_i$		
	$s_{i+1-n}t_{i+1-n}s_{i+2-n}t_{i+2-n}s_it_i$		
	...		
	$s_{i-1}t_{i-2}s_{i-1}t_{i-1}s_it_i$		
	...		
	$s_{i+1-n}t_{i+1-n}...s_{i-1}t_{i-1}s_it_i$		

Table 3.1: Feature Template applied by DirecTL+ (Jiampojamarn et al., 2010)

Table 3.1 gives the feature template used by $\Phi(i, S, T_i, T_{i-1})$. Only indicator features are included, i.e., each feature takes on a binary value indicating its presence. The context features express letter evidence in the input string S, centered around the generator S_i of each T_i. The parameter c specifies the size of the context window. The transition features enforce the cohesion on the output side. The linear-chain features associate the context window surrounding S_i with target transitions T_{i-1} and T_i, giving an extra degree of control. Finally, the joint n-gram feature utilizes the joint information between the source and target substring pairs.

3.2 Searching

Suppose we have the weighting parameters α for the scoring model ready. Given an input string, we need to search for the highest scoring target string. Notice there is no requirement of the source string being segmented beforehand. All the possible segmentations are enumerated efficiently by the search algorithm through Dynamic Programming (DP) as shown below

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

$$Q(j, t) = \max_{t', t, j-N \leq j' < j} \alpha \cdot \phi(S^j_{j'+1}, t', t) + Q(j', t')$$

$$Q(J+1, \$) = \max_{t'} \alpha \cdot \phi(\$, t', \$) + Q(J, t')$$

$Q(j, t)$ is defined as the maximum score of the target sequence ending with target substring t, generated by the letter sequence $S_1...S_j$. ϕ describes the features

extracted from the current generator substring $S^j_{j'+1}$ of target substring t, with t' to be the last generated target substring. N specifies the maximum length of the source substring. The symbol \$ represents the start and the end of a string. Suppose the source string contains J characters, $Q(J+1, \$)$ gives the score of the highest scoring target string, which can be recovered through backtracking.

This search algorithm guarantees to give the optimal target string. However, it can be slow in cases where the number of possible target substring t'. Therefore, a beam search is also implemented to speedup the search by sacrificing the optimality as follows

$$Q(0) = 0$$

$$Q(j) = \max_{t, j-N \leq j' < j, t' \in Q(j')} \alpha \cdot \phi(S^j_{j'+1}, t', t) + Q(j')$$

where $Q(j)$ is the maximum score generated by the letter sequence $S_1 ... S_j$. Each cell $Q(.)$ also keeps track of the highest scoring target substring. For example, $t' \in Q(j')$ indicates t' is the target substring that achieves the highest score at cell $Q(j')$. Therefore $Q(J)$ gives the score of the highest scoring target string under this search. Notice the formula above assumes the beam size to be 1. Jiampojamarn et al. (2010) report no significant accuracy decrease by replacing the exact search with the beam search. They also show that the larger the beam size, the better performance.

Notice both the exact search and the beam search do not insert null letters in the source string, which means DirecTL+ does not support alignments in the training data that contain null letters in the source string.

3.3 Training

The weighting parameters α are learned discriminatively in an online fashion. The Maximum Infused Relaxed Algorithm (MIRA) (Crammer and Singer, 2003) is applied for learning α. MIRA updates the weighting parameters per training instance. Given the training pair (S_i, T_i) and the weighting parameters α_{i-1} trained on the previous $i-1$ instances, this update process can be described as the following optimization problem

$$\alpha = \min_{\alpha_i} \|\alpha_i - \alpha_{i-1}\|$$

$$S.T. \;\; \alpha_i \cdot (\Phi(S_i, T_i) - \Phi(S_i, \hat{T})) \geq loss(T_i, \hat{T}) \quad \forall \hat{T} \in T_n$$

where T_n is a list of n-best outputs found under the current model parameterized by α_{i-1}. We can see this update process finds the smallest change in the current weights α_{i-1} so that the new weights α_i will separate the correct answer from each incorrect answer by a margin determined by the loss function $loss(T_i, \hat{T})$. The loss function computes the Levenshtein distance between T_i and $\hat{T}$.

Since MIRA training requires the n-best answers T_n, both beam search and exact search algorithms are modified to keep track of the n-best target substrings at each cell $Q(.)$ so that the n-best lists can be recovered from backtracking.

Chapter 4

Joint Consideration of Spellings and Supplemental Information

In this chapter, I present the approach to machine transliteration by jointly considering orthography and supplemental information. I first show how DirecTL+ could be modified to enable joint consideration of the spelling and the phonetic transcription of a given source word. Then, I will discuss the issue of aligning triple strings and how it affects the implementation of the proposed model. Next I show how to generalize the above approach to jointly considering the spelling and arbitrary number of supplemental strings, such as supplemental transliterations, of a given source word. After describing the model, I introduce additional ideas beyond the proposed model that could further boost the transliteration performance. The experiments and discussion about the results are presented at the end of this chapter.

4.1 Joint Consideration of Spellings and Phonemes

Phonetic transcriptions are in the form of phoneme strings. Thus, each training instance contains a source string, a target string and an optional phoneme string. The phoneme string is optional since transcriptions of some words are not available. Let us assume there exists consistent character-level alignments between three strings as shown in Figure 4.1. Below is how I modify each components of DirecTL+ to enable joint consideration of the spelling and the phoneme string of the source word.

4.1.1 Scoring Model

The scoring model is extended to compute a linear combination of features from three aligned strings $\alpha \cdot \Phi(S, T, P)$, where P is the phoneme string. Assume there are m aligned substrings, such that (S_i, P_i) generates T_i. Following the same Markov

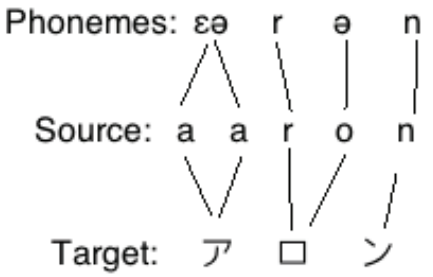

Figure 4.1: The triple alignment between the English word *aaron* ,the Japanese Katakana ア *(A)* □ *(RO)* ン *(N)* and the phoneme string */eərən/*

order assumption on the output side, the scoring model is then rewritten as

$$\sum_i \alpha \cdot \Phi(i, S, P, T_i, T_{i-1})$$

Besides the features from S and T, I extract three sets of features between P and T, i.e., context n-gram features, linear-chain features, joint n-gram features. Table 4.1 shows the feature templates for P and T. Since there is no joint feature between S and P [1], the scoring model can be further rewritten as

$$\sum_i \alpha \cdot [\Phi(i, S, T_i, T_{i-1}), \Phi'(i, P, T_i, T_{i-1})]$$

4.1.2 Searching

Given an unsegmented source string S and its unsegmented phoneme string P, I need to search for the highest-scoring target string. The original DP-based search algorithm can be generalized to achieve this goal by introducing an additional dimension to represent the segmentation of the phoneme string P

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

$$Q(j, k, t) = \max_{t', t, j-N \le j' < j, k-N' \le k' < k} \alpha \cdot [\phi(S_{j'+1}^{j}, t', t), \phi'(P_{k'+1}^{k}, t', t)] + Q(j', k', t')$$

$$Q(J+1, K+1, \$) = \max_{t'} \alpha \cdot [\phi(\$, t', \$), \phi'(\$, t', \$)] + Q(J, K, t')$$

$Q(j, k, t)$ is defined as the maximum score of the target sequence ending with the target substring t, generated by the letter sequence $S_1...S_j$ and the phoneme sequence $P_1...P_k$. ϕ' describes the features extracted from the current phoneme substring $P_{k'+1}^{k}$ that generates target substring t, with t' being the last generated target substring. ϕ remains the same as that in the original DirecTL+. Suppose the

[1] I did experiments showing that joint features from (S, P) do not help.

context	$p_{i-c}t_i$	linear-chain	$p_{i-c}t_{i-1}yt_i$
	...		...
	$p_{i+c}t_i$		$p_{i+c}t_{i-1}t_i$
	$p_{i-c}p_{i-c+1}t_i$		$p_{i-c}p_{i-c+1}t_{i-1}t_i$
	...		...
	$p_{i+c-1}p_{i+c}t_i$		$p_{i+c-1}p_{i+c}t_{i-1}t_i$
	...		...
	$p_{i-c}...p_{i+c}t_i$		$p_{i-c}...p_{i+c}t_{i-1}t_i$
joint n-gram	$p_{i+1-n}t_{i+1-n}p_it_i$		
	...		
	$p_{i-1}t_{i-1}p_it_i$		
	$p_{i+1-n}t_{i+1-n}p_{i+2-n}t_{i+2-n}p_it_i$		
	...		
	$o_{i-1}t_{i-2}p_{i-1}t_{i-1}p_it_i$		
	...		
	$p_{i+1-n}t_{i+1-n}...p_{i-1}t_{i-1}p_it_i$		

Table 4.1: Feature template for extracting features between the phoneme string P and the target string T

source string contains J characters and the phoneme string contains K phonemes, $Q(J+1, K+1, \$)$ gives the score of the highest scoring target string.

Similarly, the beam search can also be modified for the same purpose.

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

$$Q(j,k) = \max_{t, j-N \leq j' < j, k-N \leq k' < k, t' \in Q(j',k')} \alpha \cdot [\phi(S^j_{j'+1}, t', t), \phi'(P^k_{k'+1}, t', t)] + Q(j', k')$$

where $Q(j,k)$ is the maximum score generated by the letter sequence $S_1...S_j$ and the phoneme sequence $P_1...P_k$. $Q(J,K)$ gives the score of the highest scoring target string under this search.

We can see the time complexity of this modified DP-based search and beam search is changed to their original complexity of two strings multiplied by the length of the phoneme string K. This increase in time complexity becomes impractical when it comes to a large amount of training data. In Section 4.2, I will discuss how this increase can be alleviated by enforcing certain alignment between the source word and the phoneme string.

Just like the original search algorithms, the modified ones do not insert nulls in the source word nor the phoneme string, which means the training alignments that contain null letters in the source word or the phoneme string are not supported.

4.1.3 Training

The same MIRA training is applied to update the weight parameters α per training instance (S_i, P_i, T_i):

$$\alpha = \min_{\alpha_i} \|\alpha_i - \alpha_{i-1}\|$$

$$S.T.\ \ \alpha_i \cdot (\Phi(S_i, P_i, T_i) - \Phi(S_i, P_i, \hat{T})) \geq loss(T_i, \hat{T}) \quad \forall \hat{T} \in T_n$$

Since the search algorithms enumerate all the possible segmentations of both S and P, it is very likely that most outputs $\hat{T}$ in the top-n list T_n are identical but with different segmentations of S_i and P_i. This hinders the discriminative training from moving the model towards the correct outputs and away from the incorrect ones. Therefore, I need to increase the size of the top-n list, which leads to an increase in training time.

4.2 Alignment and Implementation

Three pairwise alignments, i.e., $A(S,T)$, $A(S,P)$ and $A(P,T)$ are defined as consistent alignments if they satisfy the following constraint: if S_i is aligned to T_j in $A(S,T)$ and S_i is aligned to P_k in $A(S,P)$, then P_k must be aligned to T_j in $A(P,T)$.

In order to train our model, we need to obtain triple alignments between the source word S, the phoneme string P and the target word T. A straightforward idea is to generate three pairwise alignments first and then merge them. This idea requires the three pairwise alignments to be consistent. However, I find cases where consistent pairwise alignments do not exist although each of them is precise. Let us check the example in Figure 4.2 where *abbey* is the source English word with its phoneme string */abi/* and アベイ is the target Japanese word. Figure 4.2 provides three pairwise manual alignments between each pair. The English letter *e* is aligned to the phoneme to */i/* in the first alignment and aligned to ベ*(BE)* in the second alignment. However the phoneme */i/* is aligned to イ *(I)* rater than ベ*(BE)* in the last alignment. This inconsistency is due to the fact that names are sometimes transliterated based on orthography rather than their sounds.

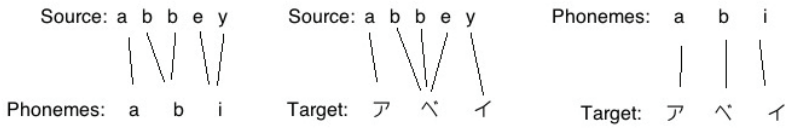

Figure 4.2: Three pairwise alignments between the English word *abbey*, the phoneme string */abi/* and the Japanese Katakana ア*(A)*ベ*(BE)*イ *(I)*

Although consistent pairwise alignments between triple strings may not exist, we can enforce triple alignments in order to train the model by abandoning one pairwise alignment. In the following two subsections, I will show how to achieve triple alignments by pivoting on either the target word or the source word respectively.

4.2.1 Pivoting on the target language

I align the source word and the phoneme string to the target word respectively, and then merge the two resultant pairwise alignments based on the target word to achieve a triple alignment. Specifically, I apply M2M-aligner to get a many-to-one pairwise alignment between the source word and the target word, and many-to-one pairwise alignment between the source phoneme string and the target word. Since the maximum length of the aligned substrings in the target word is fixed to one, these two pairwise alignments can be easily merged according to their overlaps on the target word. Figure 4.3 gives an example of how this process works. In this example, the aligned substring *bbe* in the source word and the aligned substring */b/*in the phoneme string are aligned to the same target substring ベ, thus *bbe* and */b/* is considered as aligned to each other.

However, if the target word is longer than the source word or the phoneme string, there will be null letters in the source word or the phoneme string, which are not supported by either DirecTL+ or my modifications. Therefore, I have to abandon these alignments, resulting in less training data. I can train the model described in Section 4.1 directly using the resultant triple alignments and apply the modified search algorithms to search for highest-scoring target word given an unaligned pair of a source word and a phoneme string.

Figure 4.3: Achieve triple alignment by pivoting on the target word between the source English word *abbey*, the phoneme string */abi/* and the target Japanese Katakana *ア(A)ベ(BE)イ(I)*

.

4.2.2 Pivoting on the source language

Similarly, I align the source word to the phoneme string and the target word respectively, and then achieve a triple alignment by merging the two resulted pairwise alignments. Specifically, I apply M2M-aligner to get the pairwise many-to-many

alignment between the source word and the target word, and the pairwise one-to-many alignment between the source word and the corresponding phoneme string. Then I merge these two pairwise alignments by pivoting on the source word. Figure 4.4 provides an example of this process. In this example, the substring ب in the target word and the substring in the phoneme string */b/* are aligned to the same substring *b* in the source word, thus ب and */b/* are treated as aligned to each other. These resultant triple alignments would not contain null letters in the source words regardless of the length of the target words because they are many-to-many aligned. However, there will be null letters in the phoneme strings in cases where the phoneme strings are shorter than the source words, which is not supported by the model described in Section 4.1. I make further modifications to the search algorithms so that null letters are allowed in the aligned phoneme strings.

The new search algorithms assume the source word and the phoneme string is one-to-many aligned. It only needs to search through all the possible segmentations on the source word to generate the target word. The triple alignment is then automatically achieved due to the one-to-many alignments between the source word and phoneme string. Below is the new exact search algorithm

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

$$Q(j, t) = \max_{t', t, j-N \leq j' < j} \alpha \cdot [\phi(S^j_{j'+1}, t', t), \phi'(P^k_{k'+1}, t', t)] + Q(j', t')$$

$$Q(J + 1\$) = \max_{t'} \alpha \cdot [\phi(\$, t', \$), \phi'(\$, t', \$)] + Q(J, t')$$

where $P^k_{k'+1}$ is the corresponding aligned phoneme substring of $S^j_{j'+1}$ in the one-to-many pairwise alignment between the source word and the phoneme string. The beam search version is as follows:

$$Q(0) = 0$$

$$Q(j) = \max_{t, j-N \leq j' < j, t' \in Q(j')} \alpha \cdot [\phi(S^j_{j'+1}, t', t), \phi'(P^k_{k'+1}, t', t)] + Q(j')$$

where the notation $P^k_{k'+1}$ is the same as that in the exact search. The time complexity of these two search algorithms is independent of the length of the phoneme string, which is the same as the original DirecTL+ model. Moreover, because I only search the possible segmentations on the source strings, the issues of disallowing null letters in the phoneme string and requiring a larger n-best list as discussed in Section 4.1.3 are resolved.

4.3 Generalization for Jointly Considering Arbitrary Number of Strings

Since the supplemental information to leverage can be transliterations from other languages, each training instance is then represented as $(S, T, \{P_1, ..., P_k\})$, where

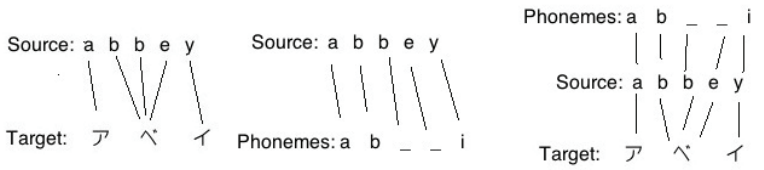

Figure 4.4: Achieve triple alignment by pivoting on the source word between the source English word *abbey*, the phoneme string */abi/* and the target Japanese Katakana *ア(A)ベ(BE)イ(I)*

.

$\{P_1, ..., P_k\}$ is a list of supplemental transliterations of (S, T) in other languages. The size of the list is not fixed because we may not have access to transliterations in certain languages for certain source words. The model described above can be easily extended to jointly consider multiple strings as follows:

- **Scoring Model** The scoring model for $(S, T, \{P_1, ..., P_k\})$ is as follows

$$\sum_i \alpha \cdot [\Phi(i, S, T_i, T_{i-1}), \Phi'(i, P_1, T_i, T_{i-1}), ..., \Phi'(i, P_k, T_i, T_{i-1})]$$

  The feature template Φ' is the same as the one described in Section 4.1.1.

- **Searching and Alignment** If I generalize the searching described in Section 4.1.2, the resultant time complexity would expand exponentially in the number of supplemental strings. Therefore, I generalize the idea in Section 4.2.2 to achieve alignment among multiple strings, and thus maintain the time complexity of searching to be independent of the number of supplemental strings.

- **Training** Naturally, the training is then modified as follows.

$$\alpha = \min_{\alpha_i} \|\alpha_i - \alpha_{i-1}\|$$

$$S.T.\ \ \alpha_i \cdot (\Phi(S_i, \{\boldsymbol{P_j}\}, T_i) - \Phi(S_i, \{\boldsymbol{P_j}\}, \hat{T})) \geq loss(T_i, \hat{T}) \quad \forall \hat{T} \in T_n$$

  where $\{\boldsymbol{P_j}\}$ is the list of supplemental transliterations.

4.4 Beyond the Joint Model

4.4.1 Language-specific preprocessing

Characters in some languages, for example Chinese and Japanese, are syllabic rather than phonemic. Therefore, these languages have large sets of characters, which increases the sparsity of the data for alignment and transliteration learning.

If we apply M2M-aligner on a sparse dataset, the resultant alignments are generally poor and meaningless. Even languages with phonemic orthography have irregular symbols that correspond to multiple phonemes, e.g., the symbol *x* in English is often pronounced as *ks*. Figure 4.5 gives an example of the alignment between an English word, its phoneme string and the corresponding Japanese Katakana. It is generated by pivoting on the source word. We can see the English letter *x* that is supposed to align to ク (*KU*) is aligned to the null letter in the pairwise alignment between English and Japanese Katakana. Another awkward part of this alignment is that although *c* is not pronounced, it is somehow aligned to a Katakana symbol.

My solution is to design language-specific methods to decompose syllabic symbols or irregular symbols into phonemic symbols. I design such methods for English and Japanese, since they are the languages of concern in my experiments. For Japanese, I replace Katakana symbols with the corresponding romanizations. For instance, エクセル in the previous example is romanized to *E KU SE RU*. We could see ク (*KU*) and ル (*RU*) share the same vowel *U* and エ (*E*) and セ (*SE*) share the same vowel *E*. Then I treat each symbol in the romanized form as an individual symbol rather than as a part of a bigger symbol. In this way the sparsity problem is alleviated. The size of the original Katakana alphabet is reduced from ∼50 to ∼20. Figure 4.6 shows the resulting alignment by romanizing the Katakana symbols in Figure 4.5. The new alignment makes more sense. However, in the resultant triple alignment, the phoneme */s/* is aligned to *K:U* but not *S*. This is because the English letter *x* does not correspond to a single phoneme but two phonemes, i.e., *k* and *s* and *x* is aligned to *K:U*. Therefore, in order to solve this issue, I replace the letter *x* in English with the letters *k* and *s*. Figure 4.7 shows the resultant alignment by replacing *x* with *k* and *s* of the example in Figure 4.6. We can see the triple alignment makes more sense.

One side-effect of this preprocessing is the loss of orthographic information. I will discuss this later in the experiment section.

4.4.2 Combining joint model with re-ranking

The re-ranking approach could boost the accuracy of the baseline system [2] but is unable to improve the recall[3] This prevents further improvement. Meanwhile, the development experiment of applying the joint model[4] shows promising improvement in the recall with relatively small improvement in accuracy over the baseline system. This suggest I should re-rank the result of the joint model, because the increase in recall brought by the joint model will give the re-ranker extra room to improve the accuracy. Therefore, this combination would presumably achieve the highest accuracy while preserving improvement in recall of the joint model. I apply

[2]Otherwise specified, baseline system refers to DirecTL+ without utilizing supplemental information.

[3]Recall is the fraction of n-best lists that contain the correct transliterations.

[4]I refer to my joint model as the model described from Section 4.1 to Section 4.3.

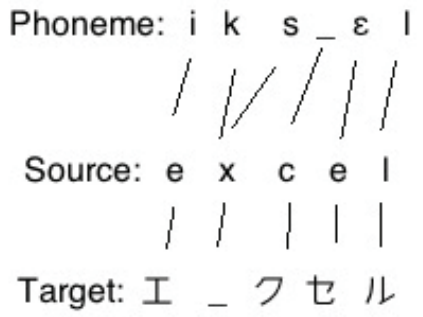

Figure 4.5: Poor triple alignment by pivoting on the source word between the source English word *excel*, the phoneme string */iksel/* and the target Japanese Katakana *エ(E)ク(KU)セ(SEル(RU)*

.

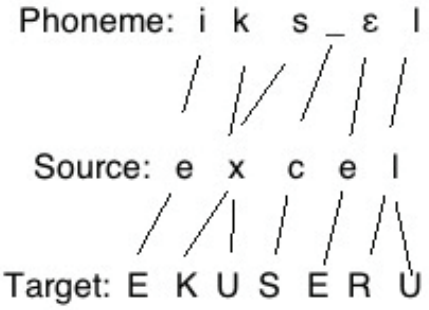

Figure 4.6: Better triple alignment by romanizing Japanese characters

.

exactly the same re-ranking algorithm described in (Bhargava and Kondrak, 2011) except that my joint model is the base system for re-ranking.

4.4.3 Apply target-language lexicons

During the development phase, I found a significant number of the wrong transliterations generated by the joint model are actually very close to the correct ones. These wrong transliterations are not valid words in the target language and are only a few letters different from the correct ones. Cherry and Suzuki (2009) report improvement in Japanese-to-English transliteration accuracy by utilizing a target-language lexicon. Therefore, I propose to re-rank the outputs of the joint model with a target-language lexicon.

A target-language lexicon is built from a monolingual target-language corpus.

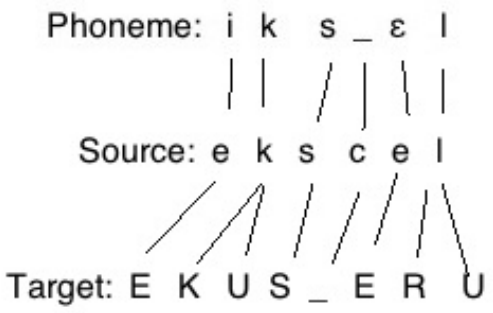

Figure 4.7: Better triple alignment by romanizing Japanese characters and converting *x* to *ks*

.

It is a list of words together with their frequencies collected from the corpus. Following Cherry and Suzuki (2009), I encode the lexicon frequency information of a given word into binary features according to coarse bins. Specifically, I chose 5 frequency bins: $[< 2000]$, $[< 200]$, $[< 20]$, $[< 2]$, $[< 1]$. So a word with the frequency 194 will cause the features $[< 2000]$ and $[< 200]$ firing. The re-ranking algorithm is the same as described in (Bhargava and Kondrak, 2011) except that the features are extracted from the target-language lexicon.

4.5 Experiment and Discussion

Similar to the experimental set up in (Bhargava, 2011), I investigate the following two tasks:

- Transliteration with phonetic transcriptions. Specifically, we conduct experiments on English-to-Japanese transliteration with phonetic transcriptions for English.
- Transliteration with supplemental transliterations for other languages. Specifically, I investigate English-to-Japanese and Japanese-to-English transliteration with supplemental transliterations from nine other languages.

4.5.1 Data

I use the phoneme strings provided in the Combilex English transcription lexicon (Richmond et al., 2009). By discarding all the diacritics and multiple-word entries, the resultant lexicon contains 113,999 entries. I also remove the stress and syllable symbols from each phoneme string.

Language	Size	Overlap
Bengali	13,624	2,152
Chinese	40,214	14,056
Hebrew	10,501	3,997
Hindi	13,427	2,507
Japanese	28,013	28,013
Kannada	11,540	2,170
Korean	7,778	7,773
Persian	12,368	4,047
Tamil	11,642	2,205
Thai	28,932	10,378

Table 4.2: The size of each pairwise NEWS dataset and the amount of common data with the English-Japanese dataset. The source language of all datasets is English.

The transliteration data is from the transliteration shared task of the 2010 Named Entities Workshop (Li et al., 2010). There are ten pairwise transliteration datasets between English and each of ten other languages, including Japanese. The size of each pairwise dataset and its overlap with English-Japanese dataset can be found at Table 4.2

I implement the idea of utilizing a target-language lexicon in Japanese-to-English transliteration experiment. The English lexicon is derived from the English gigaword monolingual corpus(LDC2012T21). I first lowercase this corpus, and then compute the unigram frequency counts. The resulting lexicons contain 7,466,441 words.

All the experiments use 80% of the dataset for training, 10% for development, 10% for held-out testing. The development set is used to tune parameters. The final result is obtained on the held-out test set by combining the training and development set as the final training set.

4.5.2 Improving Transliteration with Phonetics

As stated in Chapter 1, the phonetic transcription can help pronounce the source word, thus is helpful for transliteration. I conduct experiments on English-to-Japanese transliteration with the phonetic transcriptions for English. The dataset I use is generated by providing each entry in the NEWS10 English-to-Japanese pairwise dataset the corresponding English phoneme string, if it is available in the Combilex lexicon. Thus each entry in the resultant dataset consists of an English word, the corresponding Japanese transliteration and an optional phoneme string. I then split this dataset into training, development and testing. The training dataset contains 22,441 entries, 5,068 of which have phoneme strings. Following the the setting in (Bhargava and Kondrak, 2011; Bhargava and Kondrak, 2012; Bhargava, 2011), I further remove the entries which have no phoneme strings from the test and development sets, because I am only interested in words that have phoneme strings

Pivoting On	WORD	RECALL	Time Per Training Iteration
Target	67.3	87.1	3.5 hours
Source	68.0	87.5	2 hours

Table 4.3: Pivoting on the target vs. Pivoting on the source on the development set

during testing. The resulting development set has 594 entries and the test set has 626 entries.

I evaluate each model by computing the 0-1 word accuracy (WORD) and fraction of the 10-best lists that contain the correct transliterations (RECALL). The parameters that require tuning are the number of MIRA training iterations and the size the of context window for feature extraction. According to the development experiments, I set the number of MIRA training iterations to 5 and apply a context window of 11 letters on both spelling and phoneme strings.

I compare the model that implements the idea of pivoting on the target to the model that pivots on the source at Table 4.3. We can see pivoting on the source gives slightly better transliteration performance than pivoting on the target. However, it is more important to see that the computation cost of the former is much smaller than the latter, which put me in favour of the former idea. Therefore, I will always refer to the joint model as the one pivoting on the source in the following experiments.

I use two baseline models for comparison: the original DirecTL+ model that cannot utilize supplemental phoneme strings, and the re-ranking model that uses DirecTL+ as its base system. Both the DirecTL+ model[5] and the re-ranking model[6] are available online. I tuned them on the development set as well. Table 4.4 compares my joint models to these two baselines on the development set. My model outperforms both baseline systems, especially in terms of RECALL. Table 4.4 also shows the effect of the language-specific preprocessing of English and Japanese with the joint model. We can see the preprocessing further boosts the performance of the joint model. However, the improvements are not as significant as I expected. This is probably due the large amount of training data, which alleviates the problems caused by the sparsity of symbols and the irregular symbols. The best performance is achieved by combining re-ranking with the joint model. The increase in RECALL brought by the joint model provides extra room for the re-ranking model to push the accuracy.

Table 4.5 presents the results on the test dataset. The re-ranking approach and the joint model achieve comparable word accuracy, but the latter has consistently higher RECALL. The combination approach takes advantage of the RECALL improvement, resulting in 2 percent WORD improvement over the re-ranking approach.

[5]https://code.google.com/p/directl-p/

[6]http://www.cs.toronto.edu/ aditya/g2p-tl-rr/

Approach	Preprocessing	WORD	RECALL
DirecTL+	None	61.3	84.7
Re-ranking	None	66.5	84.7
JointModel	None	68.0	87.5
	x →k,s	68.4	87.4
	Katakana Romanization	68.4	86.7
	Both	68.4	87.5
Combination	None	68.7	87.5

Table 4.4: English-to-Japanese transliteration results on the development set. DirecTL+ is the baseline that cannot utilize phoneme strings. Re-ranking is the re-ranking approach that uses supplemental phonemes to re-rank the output of DirecTL+. JointModel is the new model presented in this thesis. Combination uses supplemental phonemes to re-rank the output of JointModel. The fourth row to the sixth row present the results of applying language-specific preprocessing.

Approach	WORD	RECALL	Training Time
DirecTL+	63.1	86.4	8.5 hours
Re-ranking	67.7	86.4	88.5 hours
JointModel	67.4	89.6	10.0 hours
Combination	70.0	89.6	103.5 hours

Table 4.5: English-to-Japanese transliteration results on the test set. The last column provides the time spent on training each model.

# of Sup TLs	0	1	2	3	4	5	6	7	8	9
# of data entries	9,516	5,628	5,396	3,056	1,754	1,002	654	547	315	145

Table 4.6: The amount of data entries in English-Japanese dataset having certain number of supplemental transliterations (Sup TLs)

4.5.3 Improving Transliteration with Supplemental Transliterations

In this task, I attempt to improve transliteration by jointly considering the spelling of the source word and supplemental transliterations from other languages. Specifically, I conduct experiments on English-to-Japanese and Japanese-to-English transliteration with supplemental transliterations from nine other languages. The training, development and test datasets are generated similarly as the last experiment. Each entry in the training may have at most nine supplemental transliterations, or may not have any supplemental transliteration at all. Each entry in the development and testing set has at least one supplemental transliterations. The sizes of the resultant training, development testing set are 22,411 , 2,289 and 2,801, respectively. Table 4.6 gives the amount of data entries having a certain number of supplemental transliterations. There are 18,497 entries for which at least one transliteration from a non-Japanese language is available.

The baselines models in this experiment for comparison are the same as the two in the previous experiment. I also implement the idea of incorporating a target lexicon as described Section 4.4.3. Table 4.7 presents the development results of both English-to-Japanese and Japanese-to-English transliteration. We can see the joint model outperforms both baselines at all evaluation metrics. In English-to-Japanese transliteration, the improvement of WORD is not big, i.e., 1 percent over the re-ranking model, but the RECALL improvement is around 3 percents, which provides more room for the re-ranking model to improve. Therefore, the combination idea further pushes WORD, resulting at 2 percents improvement over the re-ranking model. In Japanese-to-English transliteration, the improvement is much more impressive. The joint model almost doubles the WORD of the DirecTL+ model and achieves 8 percents improvement over the re-ranking approach. The RECALL improvement is 20 percents, which allows the combination idea to bring another 5 percents improvement of WORD.

Table 4.8 presents the results of incorporating a target-language lexicon on the development set. Since I only have a lexicon for English, I only present the result of incorporating a target lexicon for Japanese-to-English transliteration. We can see the English lexicon provides a significant boost to the WORD for all the models. The joint model with the combination idea and the lexicon idea gives the best performance of Japanese-to-English transliteration, which is 20 percent improvement

Approach	English-to-Japanese		Japanese-to-English	
	WORD	**RECALL**	**WORD**	**RECALL**
DirecTL+	52.7	80.6	22.2	51.6
Re-ranking	56.5	80.6	32.9	51.6
JointModel	57.6	83.1	40.8	72.4
Combination	58.5	83.1	45.6	72.4

Table 4.7: Results of utilizing supplemental transliteration for English-to-Japanese and Japanese-to-English transliteration on the development set

Approach	WORD
DirecTL+	34.5
Re-ranking	38.7
JointModel	50.1
Combination	53.6

Table 4.8: Results of incorporating an English lexicon for the Japanese-to-English transliteration on the development set

of WORD over the re-ranking model in Table 4.7.

The difference of improvements between English-to-Japanese and Japanese-to-English transliteration is probably due to the following two reasons: First, Japanese-to-English transliteration is more difficult than English-to-Japanese transliteration (the DirecTL+ model has WORD of 52.7 in the former task but only 22.2 in the latter), which provides a bigger room for improvement. This difficulty is due to the fact that most of names in our dataset are originally English names. Converting these names back from Japanese into English is challenging because their Japanese versions introduce extra vowels and removing these vowels is less predictable than introducing them. Second, because the supplemental transliterations are actually collected by pivoting on English, they presumably provide more information about the English transliterations.

Table 4.9 presents the results of the joint model and baseline models on the testing data. Most of our observations of the development dataset hold true in this dataset. One exception is that our joint model does not outperform the re-ranking model in English-to-Japanese transliteration.

4.5.4 Discussion

My joint model is shown to be more effective than the re-ranking approach in terms of utilizing supplemental information for transliteration. This is mainly due to its ability to consider the supplemental information during the transliteration generation process. This not only leads to improvement in recall, but sometimes improvement in accuracy. The former means extra room for the re-ranking approach to improve. I thereby further push the accuracy by re-ranking the outputs of the joint

Approach	English-to-Japanese		Japanese-to-English	
	WORD	**RECALL**	**WORD**	**RECALL**
DirecTL+	51.5	79.5	19.7	48.6
Re-ranking	56.8	79.5	30.3	48.6
JointModel	56.4	81.6	38.8	72.0
Combination	57.0	81.6	44.6	72.0
+Lexicon	N/A	N/A	53.1	72.0

Table 4.9: Results of utilizing supplemental transliterations for English-to-Japanese and Japanese-to-English on the test set

model. Besides the accuracy and recall scores, the joint model costs less training time than the re-ranking model does.

The idea of language-specific preprocessing, however, is not very effective. The reason is probably that the sparsity of symbols is not a big issue when it comes to a large training set, e.g., 20K instances. This may also suggest that the alignments that look poor to human does not necessarily lead to negative transliteration results.

By introducing a lexicon of the target language, I am able to achieve additional improvements. However, we should notice this only applies to cases where the target transliterations are in the target lexicon.

Chapter 5

Transliteration with Interlingua as Supplemental Information

In the previous chapter, I show both the joint model and the re-ranking model improve the transliteration by utilizing supplemental transliterations from other languages. However, because both approaches draw features directly from the orthography of all the supplemental languages, they have the following limitations:

- The feature space expands linearly in the number of supplemental languages. In the re-ranking model and the joint model, I extract features by paring each supplemental transliteration with the target transliteration. However, if I draw feature from all possible pairs, the number of features will increase exponentially.
- The feature weight learned for a character sequence of one language is not shared to another character sequence in the same or other languages that has similar sound. For example, say Japanese and Chinese are supplemental languages and English is the target language. Suppose the model has learned a feature weight for the Japanese character ク*(Ku)* co-occurring with the English letter *k*. During testing, suppose the available supplemental transliterations are Japanese characters キュ*(Kju)* and Chinese character 庫*(Ku)*, the model cannot utilize the feature weight learned for ク*(Ku)*, although all these characters sound similar and are strong indications of the target English letter to be *k*.
- As a consequence of the last limitation, both the re-ranking model and the joint model can only utilize transliterations from languages that are available during its training. However, under a realistic setting, we may have access to transliterations from new languages after the training is done. One solution is to retrain the model if we have new languages. But it is possible that the amount of transliterations from new languages are not enough for retraining.

In order to overcome these limitations, I propose to use a phonetic interlingua to represent the supplemental transliterations from all other languages, and then

extract features from this interlingua representation instead of orthography. In this way, the size of the feature space will be independent of the number of the languages, and information of phonetically-similar symbols across languages(including new languages that are not available during training) will be shared, assuming we know how to convert transliterations from each language into the interlingua representation.

This chapter presents three approaches to inducing an phonetic interlingua for transliteration generation, followed by the experiments and discussions.

5.1 IPA as Interlingua

The most straightforward way to create a phonetic interlingua is to apply the International Phonetic Alphabet (IPA). IPA is an alphabetic system of phonetic notations of the sounds of oral language. It is widely applied to represent phonetic transcriptions of many languages. This approach consists of the following steps:

1. For each supplemental language, collect a transcription lexicon. Note the transcription of each word should be represented by IPA.

2. Train a Grapheme-to-Phoneme (G2P) converter for each supplemental language based on the collected transcription lexicon. I use DirecTL+ as the G2P converter since it achieves the state-of-the-art G2P performance for several languages (Jiampojamarn et al., 2010).

3. Apply the G2P converters to convert all the supplemental transliterations into IPA strings.

4. Apply the joint model presented in Chapter 4 to learn and generate transliterations with the supplemental information to be the generated IPA strings.

Figure 5.1(b) gives an example of this interlingua representation.

5.2 English as Interlingua

English is the most widely-used language in the world. It can be treated as the ad-hoc interlingua, because names are usually transliterated into English first, and then introduced into other languages. Besides, one sacrifice of using IPA as the interlingua is the loss of orthographic information of supplemental transliterations. Thus I hope I can somehow preserve the orthographic information through English.

This approach consists of the following steps:

1. For each involved supplemental language, collect the pairwise transliteration data with English.

2. Train a transliteration model using DirecTL+ for each supplemental language.

3. Apply transliteration models to convert all the supplemental transliterations into English.

4. Apply the joint model presented in Chapter 4 to learn and generate transliterations with the supplemental information to be the generated English strings.

Figure 5.1(c) gives an example of this interlingua representation. Notice I do not aim to convert the supplemental transliterations into the correct English correspondences. Instead, I want to introduce variants that are not necessarily correct English transliterations but preserves the phonetic and orthographic information of the original supplemental transliterations.

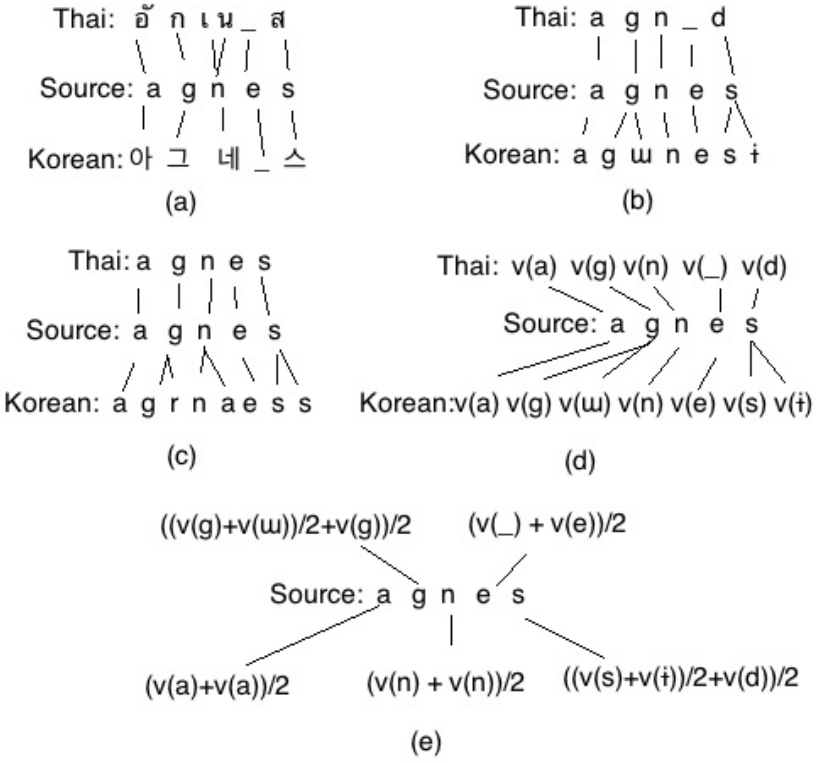

Figure 5.1: Different interlingua representations. Korean and Thai are the supplemental languages. $v(.)$ is the phonetic feature vector of a given IPA symbol. (a) The original orthographic representations. (b) IPA as interlingua (c) English as interlingua (d) Phonetic feature vectors as interlingua (e) Merged phonetic feature vector as interlingua

5.3 Phonetic feature vector as Interlingua

One drawback of using IPA as interlingua is its ignorance of the similarity between sounds[1]. This leads to two negative consequences. First, my model is not aware that the consonant /b/ is closer to the consonant /p/ than to the vowel /o/. The second and also the major negative consequence is that my model cannot utilize new IPA symbols that are not seen during its training. My solution is to convert every IPA symbol into a phonetic feature vector, each dimension of which represents its value of a phonetic feature. I follow the phonological feature chart created by Jason Riggle [2] for the conversion. Figure 5.2 (a) shows the phonetic feature vector for the phoneme */p/*. In practice, I convert this raw phonetic feature vector into a binary-valued vector as shown in Figure 5.2 (b). This binary-valued representation outperforms the raw representation in our preliminary experiments [3].

The steps in this approach are the same as the steps described in Section 5.1, except that I convert each IPA symbol to the corresponding feature vector (as shown in Figure 5.1 (d)) and I modify the joint model so that it can use the phonetic feature vectors as the supplemental information.

[labial : 1, dorsal : -1, ATR : 0, ...]

(a)

[labial+ : 1, labial- : 0, dorsal+ : 0, dorsal- : 1, ATR+ : 0, ATR- : 0, ...]

(b)

Figure 5.2: The phonetic features of the phoneme */p/* and the actual feature vector of */p/* for learning.

5.3.1 Merge Phonetic Feature Vectors

Figure 5.1 (d) provides an example of using phonetic feature vectors as interlingua, which is generated by simply replacing each IPA symbol in Figure 5.1(b) with the corresponding feature vector. Since I am interested in creating a phonetic representation that cannot be pronounced but conveys the pronunciations of the same word across different languages, I perform the following two successive merging processes before proceeding to training the modified joint model.

[1] I use sounds, phonemes, IPA symbols interchangeably in this thesis

[2] `https://dl.dropboxusercontent.com/u/5956329/Riggle/PhonChart_v1212.pdf`

[3] I refer to this binary-valued phonetic feature vector as phonetic feature vector in the rest of this thesis.

1. For each aligned token that has more than one feature vectors, I merge all the feature vectors inside that token into a single feature vector by taking the average. For the example in Figure 5.1 (d), the Korean phonetic feature vectors for */g/* and */ɯ/* should be merged because they are aligned to the same source letter *g*.

2. Then I merge the aligned feature vectors across the supplemental languages into a single feature vector by taking the average.

Figure 5.1 (e) gives the result of merging the vectors in Figure 5.1 (d)). We can see the pronunciation of the letter *g* is in someplace between */n/* and */g/*. This is the actual form used by the modified joint model. From a practical point of view, merging vectors saves time in feature extraction, because the number of feature vectors is independent of the number of supplemental transliterations. For the example in Figure 5.1 (d), the number of vectors is 5 $\times$ the number of supplemental transliterations, while the number of vectors in Figure 5.1 (e) will always be 5 regardless of the number of supplemental transliterations.

5.3.2 Joint Model Modification

I need to modify the joint model so that it can be trained on examples as shown in Figure 5.1 (e). The feature extraction function $\Phi'()$ is the only component I need to change. I change the parameters of this function to be the merged phonetic feature vector sequence $\{\boldsymbol{P_l}\}$ and the current target symbol T_i. Suppose each phonetic feature vector $\boldsymbol{P_l}$ is a set of feature/value pairs as follows:

$$\boldsymbol{P_l} = \{(f_{lj} : v_{lj}) | 0 \leq j \leq K\}$$

where K is the number of phonetic features. Then $\Phi'()$ extracts a set of feature/value pairs from $\{\boldsymbol{P_l}\}$ and T_i.

$$\Phi'(\{\boldsymbol{P_l}\}, T_i) = \{((f_{lj}, T_i, l-i) : v_{lj}) | 0 \leq j \leq K, i-c \leq l \leq i+c\}$$

where c is the size of the context window around current vector P_l. Notice each feature in $\Phi'()$ is jointly defined by the phonetic feature f_{lj}, the target symbol T_j and the offset to the current feature vector $l-i$. Also note that the feature values are no not binary because of the merging process.

5.4 Experiments and Discussions

I evaluate my interlingua ideas on English-to-Japanese transliteration with supplemental transliterations from Hindi, Hebrew, Korean and Thai. I investigate the following two aspects of each interlingua representation:

- **Effectiveness** Whether an interlingua representation can effectively encode the supplemental transliterations to improve transliteration. Specifically, the supplemental transliterations in the training set and test set are from the same set of languages. Then I compare the models using interlingua representations to the model using orthography.

- **Multilinguality** Whether an interlingua representation is capable of utilizing transliterations from new languages. Specifically, I train the models using interlingua representations with one set of supplemental languages and test with additional supplemental languages.

5.4.1 Data

The transliteration data is the 2010 Named Entity Workshop as introduced in Section 4.5.1, which contains nine languages other than English and Japanese. I collect the transcription data for each of the nine languages from Wiktionary[4]. However, only the transcription data of Hebrew, Hindi, Korean and Thai is enough to train G2P converters. Therefore, I only consider supplemental transliterations from these four languages for the sake of comparison.

Following Section 4.5.1, I use 80% of the data for training, 10% for development and 10% for held-out test in the following experiments.

5.4.2 Effectiveness

The transliteration data is generated similarly as described in Section 4.5.3. The resultant training set contains 22,441 entries, 11,779 of which have at least one supplemental transliterations. The development and testing set contains 1,393 and 1,412 entries respectively, all of which have at least one supplemental transliteration.

Both the IPA as interlingua idea and the phonetic feature vector as interlingua idea require G2P converters to convert supplemental transliterations into IPA strings. These G2P converters are trained on the transcription data I collect from Wiktionary. In order to train effective G2P converters, I split the transcription data of each language into a training (90%) set and a development(10%) set to tune the training parameters. Then the training and development sets are joined together for the final G2P training using the tuned parameters. Table 5.1 presents the G2P performance on the development transcription set for each language. Since the size of the training data is small and the data from Wiktionary is very noisy, we can see the G2P converters perform poorly even though they are tuned to fit the development set. The G2P converter would perform presumably more poorly in converting words in the transliteration dataset, because names are usually more difficult for G2P conversion (Bhargava and Kondrak, 2012).

[4] http://en.wiktionary.org/wiki/Wiktionary:Main_Page

Language	WORD	Size of the data
Hebrew	21.3	475
Hindi	25.9	819
Korean	40.9	3,181
Thai	15.2	911

Table 5.1: G2P performance on the Wiktionary development set.

Language	WORD	Size of the data
Hebrew	24.7	9,585
Hindi	49.4	13,049
Korean	19.7	7,255
Thai	43.8	28,136

Table 5.2: The development results of applying DirecTL+ to transliterate each supplemental language into English for the purpose of using English as interlingua

Using English as interlingua requires transliteration models to convert the supplemental transliterations into English. These models are trained on the pairwise transliteration data from the 2010 NEWS using DirecTL+. Following the same training strategy as the G2P training, I tune the training parameters of each transliteration model by splitting the data into training and development sets. Table 5.2 presents the development results of transliterating each supplemental languages into English. We can see the accuracy numbers are much better than those in the G2P training except for Korean. However, unlike G2P training, I am not interested in obtaining higher accuracy. Suppose if all the models achieve 100% accuracy, then all the supplemental transliterations will be converted into the source English word; Therefore, the English-to-Japanese transliteration would not benefit from this English interlingua.

The baseline model is the DirecTL+ model that cannot leverage supplemental transliterations. I compare different representations, including the orthographic representation, of supplemental transliterations to see whether they are effective in improving English-to-Japanese transliteration. The model that uses these representations is the joint model. Table 5.3 presents the results on the development set. IPA is the only interlingua that outperforms the orthography while the other two interlingua give worse results than the orthography. Although the improvement brought by the IPA interlingua over the orthography is within one percent, it is very impressive given the poor performance of the G2P converters and its ignorance of orthographic information. This is mainly because supplemental transliterations from different languages are able to share the feature weights during the machine learning process through this common representation. The last row in Table 5.3 shows the result of using both IPAs and orthography of the supplemental transliterations. We can see further improvement in accuracy is achieved.

By looking at the English transliterations converted from the supplemental translit-

Representation	WORD	RECALL
N/A(Baseline)	54.7	82.3
Orthography	58.5	84.0
IL = IPA	59.4	84.0
IL = English	54.6	81.9
IL = Vector	56.5	83.1
IPA + Orthography	60.0	83.6

Table 5.3: Results of using different representations of supplemental transliterations on the English-to-Japanese development set. The first row is the baseline DirecTL+ model that cannot leverage supplemental transliterations.The second row is the result of using the orthographic representation of supplemental transliterations. The following three rows are the results of applying different interlingua representation (IL) ideas. The last row is the result of using both orthography and IPAs of supplemental transliterations. The transliteration model in this experiment is the joint model.

erations, I found not only the orthographic information but also the phonetic information is lost. Most of these converted English transliterations are only few letters different from the correct ones. For example, the converted supplemental transliterations for the source word *acosta* are *ecosta* (Hebrew), *acosta* (Korean), *akosta* (Thai). Therefore, they do not provide useful information for English-to-Japanese transliteration.

Although the phonetic feature vector presentation is theoretically neat, it is quite ineffective in improving transliteration. This is mainly due to three reasons. First, the phonetic features are not optimal in encoding phonetic similarities between phonemes. For example, according to these phonetic features, the most similar phoneme to /z/ is /ð/ while /ʐ/ or /ʑ/seems to be a better choice. Second, it is problematic to take the average for merging phonetic feature vectors. For example, the phoneme closest to the average of /n/ and /g/ is neither /n/ nor /g/ , but /d/. Last, as a side-effect of phonetic feature vector merging process, the original pronunciations of the supplemental transliterations are lost.

Table 5.4 presents the results on the test set. IPA representation achieves comparable performance to the orthographic representation. This confirms the poorly generated IPAs are as effective as the orthography in encoding supplemental transliterations.

5.4.3 Multilinguality

The next aspect to investigate is whether the proposed interlingua representations are capable of leveraging supplemental transliterations from new languages. Specifically, I train English-to-Japanese transliteration models with supplemental transliterations from Hebrew, Hindi and Thai, but test with optional supplemental translit-

Representation	WORD	RECALL
N/A(Baseline)	54.5	81.5
Orthography	59.6	83.6
IL = IPA	59.0	83.3
IPA + Orthography	59.6	83.1

Table 5.4: Results of using different representations of supplemental transliterations on the English-to-Japanese test set

Representation	Supplemental transliterations	WORD	RECALL
Orthography	None	58.1	84.4
	Hebrew, Hindi, Thai	59.0	84.7
IL = IPA	Hebrew, Hindi, Thai	58.8	85.7
	Hebrew, Hindi, Thai, Korean	59.7	84.8
	Korean	57.1	82.8
IL = English	Hebrew, Hindi, Thai	57.7	85.4
	Hebrew, Hindi, Thai, Korean	56.5	83.8
IL = Vector	Hebrew, Hindi, Thai	54.6	80.2
	Hebrew, Hindi, Thai, Korean	56.6	83.2

Table 5.5: Multilingual performance on the development set.The first two rows are the results of using the orthographic representation of supplemental transliterations. The rest rows are the results of applying different interlingua representation (IL) ideas. The transliteration model in this experiment is the joint model.

erations from Korean. The training set is generated by removing the Korean transliterations from the training set in the last experiment. The development/test sets are generated by removing the entries that have no Korean supplemental transliterations from the development/test sets in the last experiment.

Table 5.5 presents the results of this experiment on the development set. IPA still remains the most effective representation in terms of multilinguality. It achieves better accuracy given additional Korean transliterations. I also present of results of providing only Korean transliterations to the model with IPA interlingua. The accuracy drops and is even worse than the baseline that does not use any supplemental transliterations. This is mainly due to noisy IPAs generated from Korean transliterations.

Table 5.6 presents the results on the test set. The differences between the last four rows are slight. However, by comparing the third row and the fourth row, we can see IPA interlingua is still capable of leverage transliterations from new languages to achieve improvement in WORD and RECALL.

Representation	Supplemental transliterations	WORD	RECALL
Orthography	None	58.9	82.1
	Hebrew, Hindi, Thai	60.0	83.9
IL = IPA	Hebrew, Hindi, Thai	59.7	83.9
	Hebrew, Hindi, Thai, Korean	59.8	84.6
	Korean	59.4	84.7

Table 5.6: Multilingual performance on the test set.

5.5 Discussion

The IPA interlingua is the only effective interlingua representation according to my experiments. Although it only achieves comparable performance to the orthographic representation, the potential improvement it could bring is promising, given its ignorance of orthographic information and the poor G2P performance in my experiments. This may suggest that it is the phonetic information underneath the orthographic information of supplemental transliterations that actually helps improve transliteration.

The reason why the English interlingua and phonetic feature vector interlingua do not work is the absence of the original orthographic and phonetic information. The transliteration models, which are applied by the English interlingua approach for converting supplemental transliterations into English, tend to follow the English orthography and phonology. Thus, both the original phonetic and orthographic information are lost. The phonetic feature vector interlingua approach takes the average of the phonetic vectors across supplemental languages. This indeed creates a fuzzy and theoretically neat representation, but it fails to preserve the original pronunciations of the supplemental transliterations. The merge process also jeopardizes the phonetic feature vector's capability of leveraging phonemes that are not seen in the training data. For the example discussed earlier of merging */g/* and */n/*, suppose */g/* is the unseen IPA, the merged phonetic vector is neither approximate to */g/* nor to */n/*, but to */d/*. Therefore, the merged vector fails to encode the phonetic information from the unseen phoneme */g/*.

Chapter 6

Conclusion and Future work

In this thesis, I proposed a model that is capable of leveraging supplemental transliterations during the transliteration generation process. This model achieved not only comparable accuracy to the state-of-the-art re-ranking model, but consistently better recall. Since a better recall means a bigger for re-ranking to improve, I achieved the highest accuracy by re-ranking the output of the joint generation model.

I further explored the idea of applying phonetic interlingua to represent supplemental transliterations. Notably, this is the first application of interlingua in transliteration generation to the best of my knowledge. I investigated in particular ideas of using English, IPA and phonetic feature vectors as interlingua respectively. My experiments show only IPA achieved comparable results to orthography. However, given the poor G2P performance, I conclude that IPA has great potential in encoding supplemental transliterations. The other two interlingua representation are shown to be ineffective due to their incapability of preserving the original pronunciations of the supplemental transliterations.

6.1 Future work

Both my work in this thesis and Bhargava and Kondrak (2012) show transliteration models for a language pair can benefit transliterations for from other languages. However, the models proposed in both works cannot achieve improvement if there is no supplemental transliterations available. Therefore, I would like to eliminate the need for available supplemental transliterations. The idea is to develop a model that is able to jointly transliterate from an source language, say English, into multiple target languages, such as Japanese and Chinese. During the transliteration process, the model will generate both Japanese and Chinese transliterations, thereby it can leverage each of the two transliterations to improve the other either through re-ranking or the approach described in this work.

Regarding the interlingua representation, I would like to further push the results of using IPA as interlingua by replacing the poor IPAs with golden standard ones or applying a better G2P converter. The idea of using phonetic feature vectors is

neat, but it fails due to the merging process. Simply taking the average causes the loss of original pronunciations of supplemental transliterations. Therefore, I will investigate more elaborate and advanced techniques, such as representation learning, for the merging process. In addition, the manually-devised phonetic features are not necessarily the optimal features for transliteration generation, I will apply representation learning techniques to derive better features.

References

Nasreen Abduljaleel and Leah S. Larkey. 2003. Statistical transliteration for english-arabic cross language information retrieval. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, pages 139–146, New York, NY, USA. ACM.

M. Arbabi, S. M. Fischthal, V. C. Cheng, and E. Bart. 1994. Algorithms for arabic name transliteration. *IBM J. Res. Dev.*, 38(2):183–194, March.

Aditya Bhargava and Grzegorz Kondrak. 2011. How do you pronounce your name? improving g2p with transliterations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 399–408, Portland, Oregon, USA, June. Association for Computational Linguistics.

Aditya Bhargava and Grzegorz Kondrak. 2012. Leveraging supplemental representations for sequential transduction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 396–406, Montréal, Canada, June. Association for Computational Linguistics.

Aditya Bhargava. 2011. Leveraging supplemental transcriptions and transliterations via re-ranking. Master's thesis, University of Alberta.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguist.*, 19(2):263–311, June.

Colin Cherry and Hisami Suzuki. 2009. Discriminative substring decoding for transliteration. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1066–1075, Singapore, August. Association for Computational Linguistics.

Michael A. Covington. 1996. An algorithm to align words for historical comparison. *Comput. Linguist.*, 22(4):481–496, December.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *J. Mach. Learn. Res.*, 3:951–991, March.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38.

Wei Gao, Kam-Fai Wong, and Wai Lam. 2004. Improving transliteration with precise alignment of phoneme chunks and using contextual features. In *Proceedings of the 2004 International Conference on Asian Information Retrieval Technology*, AIRS'04, pages 106–117, Berlin, Heidelberg. Springer-Verlag.

Hotelling Harold. 1936. Relation between two sets of variables. *Biometrica*, 28:322–377.

Ulf Hermjakob, Kevin Knight, and Hal Daumé III. 2008. Name translation in statistical machine translation - learning when to transliterate. In *Proceedings of ACL-08: HLT*, pages 389–397, Columbus, Ohio, June. Association for Computational Linguistics.

Jagadeesh Jagarlamudi and Hal Daume III. 2012. Regularized interlingual projections: Evaluation on multilingual transliteration. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 12–23, Jeju Island, Korea, July. Association for Computational Linguistics.

K. S. Jeong, Sung-Hyon Myaeng, J. S. Lee, and K. Choi. 1999. Automatic identification and back-transliteration of foreign words for information retrieval. *Inf. Process. Manage.*, 35(4):523–540.

Sittichai Jiampojamarn, Grzegorz Kondrak, and Tarek Sherif. 2007. Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 372–379, Rochester, New York, April. Association for Computational Linguistics.

Sittichai Jiampojamarn, Colin Cherry, and Grzegorz Kondrak. 2008. Joint processing and discriminative training for letter-to-phoneme conversion. In *Proceedings of ACL-08: HLT*, pages 905–913, Columbus, Ohio, June. Association for Computational Linguistics.

Sittichai Jiampojamarn, Aditya Bhargava, Qing Dou, Kenneth Dwyer, and Grzegorz Kondrak. 2009. Directl: a language independent approach to transliteration. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 28–31, Suntec, Singapore, August. Association for Computational Linguistics.

Sittichai Jiampojamarn, Colin Cherry, and Grzegorz Kondrak. 2010. Integrating joint n-gram features into a discriminative training framework. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 697–700, Los Angeles, California, June. Association for Computational Linguistics.

Sung Young Jung, SungLim Hong, and Eunok Paek. 2000. An english to korean transliteration model of extended markov window. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 1*, COLING '00, pages 383–389, Stroudsburg, PA, USA. Association for Computational Linguistics.

Byung-Ju Kang and Key-Sun Choi. 2000. Automatic transliteration and back-transliteration by decision tree learning. In *Proceedings of Language Resources and Evaluation*.

Sarvnaz Karimi, Andrew Turpin, and Falk Scholer. 2006. English to persian transliteration. In Fabio Crestani, Paolo Ferragina, and Mark Sanderson, editors, *String Processing and Information Retrieval*, volume 4209 of *Lecture Notes in Computer Science*, pages 255–266. Springer Berlin Heidelberg.

Sarvnaz Karimi, Falk Scholer, and Andrew Turpin. 2007. Collapsed consonant and vowel models: New approaches for english-persian transliteration and back-transliteration. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 648–655, Prague, Czech Republic, June. Association for Computational Linguistics.

Sarvnaz Karimi, Falk Scholer, and Andrew Turpin. 2011. Machine transliteration survey. *ACM Comput. Surv.*, 43(3):17:1–17:46, April.

Brett Kessler. 2005. Phonetic comparison algorithms1. *Transactions of the Philological Society*, 103(2):243–260.

Mitesh M. Khapra, A Kumaran, and Pushpak Bhattacharyya. 2010. Everybody loves a rich cousin: An empirical study of transliteration through bridge languages. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 420–428, Los Angeles, California, June. Association for Computational Linguistics.

Mitesh M. Khapra, Raghavendra Udupa, A. Kumaran, and Pushpak Bhttacharyya. 2011. Pr + rq $\approx$rp:transliteration mining using bridge language. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intellingence*. AAAI Press, July.

Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Computational Linguisitics*, 24(4), December.

A. Kumaran, Mitesh M. Khapra, and Pushpak Bhattacharyya. 2010. Compositional machine transliteration. 9(4):13:1–13:29, December.

Phillippe Langlais. 2013. Mapping source to target strings without alignment by analogical learning: A case study with transliteration. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 684–689, Sofia, Bulgaria, August. Association for Computational Linguistics.

Haizhou Li, Min Zhang, and Jian Su. 2004. A joint source-channel model for machine transliteration. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Haizhou Li, A Kumaran, Min Zhang, and Vladimir Pervouchine. 2010. Report of news 2010 transliteration generation shared task. In *Proceedings of the 2010 Named Entities Workshop*, pages 1–11, Uppsala, Sweden, July. Association for Computational Linguistics.

Haibo Li, Jing Zheng, Heng Ji, Qi Li, and Wen Wang. 2013. Name-aware machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 604–614, Sofia, Bulgaria, August. Association for Computational Linguistics.

Wei-Hao Lin and Hsin-Hsi Chen. 2002. Backward machine transliteration by learning phonetic similarity. In *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*, COLING-02, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Comput. Linguist.*, 29(1):19–51, March.

Jong-Hoon Oh and Key-Sun Choi. 2002. An english-korean transliteration model using pronunciation and contextual rules. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Vladimir Pervouchine, Haizhou Li, and Bo Lin. 2009. Transliteration alignment. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 136–144, Suntec, Singapore, August. Association for Computational Linguistics.

Korin Richmond, Robert Clark, and Sue Fitt. 2009. Robust lts rules with the combilex speech technology lexicon. In *Proceedings of Interspeech*, pages 1259–1298, Brighton, UK, September.

Tarek Sherif and Grzegorz Kondrak. 2007. Substring-based transliteration. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 944–951, Prague, Czech Republic, June. Association for Computational Linguistics.

Tao Tao, Su-Youn Yoon, Andrew Fister, Richard Sproat, and ChengXiang Zhai. 2006. Unsupervised named entity transliteration using temporal and phonetic correlation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 250–257, Stroudsburg, PA, USA. Association for Computational Linguistics.

Raghavendra Udupa and Mitesh M. Khapra. 2010. Transliteration equivalence using canonical correlation analysis. In *Proceedings of the 32Nd European Conference on Advances in Information Retrieval*, ECIR'2010, pages 75–86, Berlin, Heidelberg. Springer-Verlag.

Su-Youn Yoon, Kyoung-Young Kim, and Richard Sproat. 2007. Multilingual transliteration using feature based phonetic method. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 112–119, Prague, Czech Republic, June. Association for Computational Linguistics.

Dmitry Zelenko and Chinatsu Aone. 2006. Discriminative methods for transliteration. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 612–617, Stroudsburg, PA, USA. Association for Computational Linguistics.

Min Zhang, Xiangyu Duan, Vladimir Pervouchine, and Haizhou Li. 2010. Machine transliteration: Leveraging on third languages. In *Coling 2010: Posters*, pages 1444–1452, Beijing, China, August. Coling 2010 Organizing Committee.