

Learning Phoneme Mappings for Transliteration without Parallel Data

Sujith Ravi and Kevin Knight
University of Southern California
Information Sciences Institute
Marina del Rey, California 90292
{sravi, knight}@isi.edu

Abstract

We present a method for performing machine transliteration without any parallel resources. We frame the transliteration task as a decipherment problem and show that it is possible to learn cross-language phoneme mapping tables using only monolingual resources. We compare various methods and evaluate their accuracies on a standard name transliteration task.

1 Introduction

Transliteration refers to the transport of names and terms between languages with different writing systems and phoneme inventories. Recently there has been a large amount of interesting work in this area, and the literature has outgrown being citable in its entirety. Much of this work focuses on *back-transliteration*, which tries to restore a name or term that has been transported into a foreign language. Here, there is often only one correct target spelling—for example, given `jyon.kairu` (the name of a U.S. Senator transported to Japanese), we must output “Jon Kyl”, not “John Kyre” or any other variation.

There are many techniques for transliteration and back-transliteration, and they vary along a number of dimensions:

- phoneme substitution vs. character substitution
- heuristic vs. generative vs. discriminative models
- manual vs. automatic knowledge acquisition

We explore the third dimension, where we see several techniques in use:

- Manually-constructed transliteration models, e.g., (Hermjakob et al., 2008).
- Models constructed from bilingual dictionaries of terms and names, e.g., (Knight and Graehl, 1998; Huang et al., 2004; Haizhou et al., 2004; Zelenko and Aone, 2006; Yoon et al., 2007; Li et al., 2007; Karimi et al., 2007; Sherif and Kondrak, 2007b; Goldwasser and Roth, 2008b).
- Extraction of parallel examples from bilingual corpora, using bootstrap dictionaries e.g., (Sherif and Kondrak, 2007a; Goldwasser and Roth, 2008a).
- Extraction of parallel examples from comparable corpora, using bootstrap dictionaries, and temporal and word co-occurrence, e.g., (Sproat et al., 2006; Klementiev and Roth, 2008).
- Extraction of parallel examples from web queries, using bootstrap dictionaries, e.g., (Nagata et al., 2001; Oh and Isahara, 2006; Kuo et al., 2006; Wu and Chang, 2007).
- Comparing terms from different languages in phonetic space, e.g., (Tao et al., 2006; Goldberg and Elhadad, 2008).

In this paper, we investigate methods to acquire transliteration mappings *from non-parallel sources*. We are inspired by previous work in unsupervised learning for natural language, e.g. (Yarowsky, 1995;

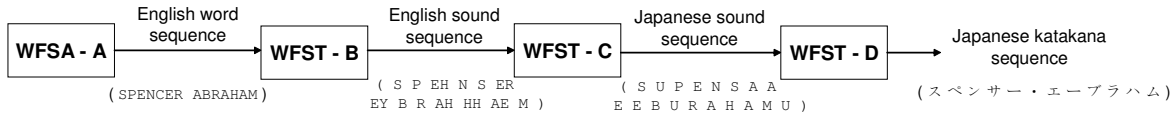


Figure 1: Model used for back-transliteration of Japanese katakana names and terms into English. The model employs a four-stage cascade of weighted finite-state transducers (Knight and Graehl, 1998).

Goldwater and Griffiths, 2007), and we are also inspired by cryptanalysis—we view a corpus of foreign terms as a code for English, and we attempt to break the code.

2 Background

We follow (Knight and Graehl, 1998) in tackling back-transliteration of Japanese katakana expressions into English. Knight and Graehl (1998) developed a four-stage cascade of finite-state transducers, shown in Figure 1.

- **WFSA A** - produces an English word sequence w with probability $P(w)$ (based on a unigram word model).
- **WFST B** - generates an English phoneme sequence e corresponding to w with probability $P(e|w)$.
- **WFST C** - transforms the English phoneme sequence into a Japanese phoneme sequence j according to a model $P(j|e)$.
- **WFST D** - writes out the Japanese phoneme sequence into Japanese katakana characters according to a model $P(k|j)$.

Using the cascade in the reverse (noisy-channel) direction, they are able to translate new katakana names and terms into English. They report 36% error in translating 100 U.S. Senators’ names, and they report exceeding human transliteration performance in the presence of optical scanning noise.

The only transducer that requires parallel training data is WFST C. Knight and Graehl (1998) take several thousand phoneme string pairs, automatically align them with the EM algorithm (Dempster et al., 1977), and construct WFST C from the aligned phoneme pieces.

We re-implement their basic method by instantiating a densely-connected version of WFST C with

all 1-to-1 and 1-to-2 phoneme connections between English and Japanese. Phoneme bigrams that occur fewer than 10 times in a Japanese corpus are omitted, and we omit 1-to-3 connections. This initial WFST C model has 15320 uniformly weighted parameters. We then train the model on 3343 phoneme string pairs from a bilingual dictionary, using the EM algorithm. EM immediately reduces the connections in the model to those actually observed in the parallel data, and after 14 iterations, there are only 188 connections left with $P(j|e) \geq 0.01$. Figure 2 shows the phonemic substitution table learnt from parallel training.

We use this trained WFST C model and apply it to the U.S. Senator name transliteration task (which we update to the 2008 roster). We obtain 40% error, roughly matching the performance observed in (Knight and Graehl, 1998).

3 Task and Data

The task of this paper is to learn the mappings in Figure 2, *but without parallel data*, and to test those mappings in end-to-end transliteration. We imagine our problem as one faced by monolingual English speaker wandering around Japan, reading a multitude of katakana signs, listening to people speak Japanese, and eventually deciphering those signs into English. To mis-quote Warren Weaver:

“When I look at a corpus of Japanese katakana, I say to myself, this is really written in English, but it has been coded in some strange symbols. I will now proceed to decode.”

Our larger motivation is to move toward easily-built transliteration systems for all language pairs, regardless of parallel resources. While Japanese/English transliteration has its own particular features, we believe it is a reasonable starting point.

<i>e</i>	<i>j</i>	$P(j e)$	<i>e</i>	<i>j</i>	$P(j e)$	<i>e</i>	<i>j</i>	$P(j e)$	<i>e</i>	<i>j</i>	$P(j e)$	<i>e</i>	<i>j</i>	$P(j e)$	<i>e</i>	<i>j</i>	$P(j e)$	<i>e</i>	<i>j</i>	$P(j e)$			
AA	o	0.49	AY	a i	0.84	EH	e	0.94	HH	h	0.95	L	r	0.62	OY	o i	0.89	SH	sh y	0.33	V	b	0.75
	a	0.46		i	0.09		a	0.03		w	0.02		r u	0.37		o e	0.04		sh	0.31		b u	0.17
	oo	0.02		a	0.03					h a	0.02					o	0.17		y u	0.12		w	0.03
	aa	0.02		i y	0.01											i	0.04		ssh y	0.12		a	0.02
				a y	0.01											sh i	0.04		ssh	0.02			
																e	0.01						
AE	a	0.93	B	b	0.82	ER	aa	0.8	IH	i	0.89	M	m	0.68	P	p	0.63	T	t	0.43	W	w	0.73
	ssh	0.02		b u	0.15		a	0.08		e	0.05		m u	0.22		p u	0.16		t o	0.25		u	0.17
	an	0.02					a r	0.03		i n	0.01		n	0.08		pp u	0.13		tt o	0.17		o	0.04
							r u	0.02		a	0.01					pp	0.06		ts	0.04		i	0.02
							o r	0.02											tt	0.03			
							e r	0.02											u	0.02			
																			ts u	0.02			
																			ch	0.02			
AH	a	0.6	CH	tch i	0.27	EY	e e	0.58	IY	i i	0.58	N	n	0.96	PAUSE	pause	1.0	TH	s u	0.48	Y	y	0.7
	o	0.13		ch	0.24		e	0.15		i	0.3		nn	0.02		s	0.22		s	0.22		i	0.26
	e	0.11		ch i	0.23		e i	0.12		e	0.07					sh	0.16		sh	0.16		e	0.02
	i	0.07		ch y	0.2		a	0.1		e e	0.03					t o	0.04		t o	0.04		a	0.02
	u	0.06		tch y	0.02		a i	0.03								ch	0.04		ch	0.04			
				tch	0.02											t e	0.02		t e	0.02			
				ssh y	0.01											t	0.02		t	0.02			
				k	0.01											a	0.02		a	0.02			
AO	o	0.6	D	d	0.54	F	h	0.58	JH	j y	0.35	NG	n	0.62	R	r	0.61	UH	u	0.79	Z	z	0.27
	oo	0.27		d o	0.27		h u	0.35		j	0.24		g u	0.22		a	0.27		u u	0.09		z u	0.25
	a	0.05		dd o	0.06		hh	0.04		j i	0.21		ng	0.09		o	0.07		u a	0.04		u	0.16
	on	0.03		z	0.02		hh u	0.02		jj i	0.14		i	0.04		r u	0.03		dd	0.03		s u	0.07
	au	0.03		j	0.02					z	0.04		u	0.01		a a	0.01		u ssh	0.02		j	0.06
	u	0.01		u	0.01								o	0.01					o	0.02		a	0.06
													a	0.01								n	0.03
																						i	0.03
																						s	0.02
																						o	0.02
AW	au	0.69	DH	z	0.87	G	g	0.66	K	k	0.53	OW	o	0.57	S	su	0.43	UW	u u	0.67	ZH	j y	0.43
	aw	0.15		z u	0.08		g u	0.19		k u	0.2		oo	0.39		s	0.37		u	0.29		j i	0.29
	ao	0.06		az	0.04		gg u	0.1		kk u	0.16		ou	0.02		sh	0.08		y u	0.02		j	0.29
	a	0.04					g y	0.03		kk	0.05					u	0.05						
	u	0.02					gg	0.01		ky	0.02					ss	0.02						
	oo	0.02					g a	0.01		ki	0.01					ssh	0.01						
	o	0.02																					

Figure 2: Phonemic substitution table learnt from 3343 parallel English/Japanese phoneme string pairs. English phonemes are in uppercase, Japanese in lowercase. Mappings with $P(j|e) > 0.01$ are shown.

A A C H I D O	C H E N J I	N E B A D A	W A K O B I A
A A K U P U R A Z A	C H E S	:	W A N K A P P U
A A N D O	:	O P U T I K U S U	W A N T E N P O
A A T I S U T O	D E K O R A T I B U	:	W A S E R I N
A A T O S E R I N A P I S U T O N	D E T O M O R U T O	P I I T A A	Y U N I O N
A I A N B I R U	E P I G U R A M U	P I K K A A	Y U N I T T O S H I S U T E M U
A I D I I D O	E R A N D O	P I N G U U	Y U U
A I K E N B E R I I	:	P I P E R A J I N A M I D O	:
A J I A K A P P U	J Y A I A N T S U	P I S A	:
A J I T O	J Y A Z U	P I U R A	Z E N E R A R U E A K O N
A K A S H I A K O O S U	:	P O I N T O	Z E R O
A K U A	M Y U U Z E U M U	:	Z O N B I I Z U
:	:	:	:
:	:	:	:
:	:	:	:

Figure 3: Some Japanese phoneme sequences generated from the monolingual katakana corpus using WFST D.

Our monolingual resources are:

- 43717 unique Japanese katakana sequences collected from web newspaper data. We split multi-word katakana phrases on the center-dot (“.”) character, and select a final corpus of 9350 unique sequences. We add monolingual Japanese versions of the 2008 U.S. Senate roster.¹
- The CMU pronunciation dictionary of English,

with 112,151 entries.

- The English gigaword corpus. Knight and Graehl (1998) already use frequently-occurring capitalized words to build the WFSA A component of their four-stage cascade.

We seek to use our English knowledge (derived from 2 and 3) to decipher the Japanese katakana corpus (1) into English. Figure 3 shows a portion of the Japanese corpus, which we transform into Japanese phoneme sequences using the monolingual resource of WFST D. We note that the Japanese phoneme inventory contains 39 unique (“ciphertext”) symbols,

¹We use “open” EM testing, in which unlabeled test data is allowed to be part of unsupervised training. However, no parallel data is allowed.

compared to the 40 English (“plaintext”) phonemes.

Our goal is to compare and evaluate the WFST C model learnt under two different scenarios—(a) using parallel data, and (b) using monolingual data. For each experiment, we train only the WFST C model and then apply it to the name transliteration task—decoding 100 U.S. Senator names from Japanese to English using the automata shown in Figure 1. For all experiments, we keep the rest of the models in the cascade (WFSA A, WFST B, and WFST D) unchanged. We evaluate on whole-name error-rate (maximum of 100/100) as well as normalized word edit distance, which gives partial credit for getting the first or last name correct.

4 Acquiring Phoneme Mappings from Non-Parallel Data

Our main data consists of 9350 unique Japanese phoneme sequences, which we can consider as a single long sequence j . As suggested by Knight et al (2006), we explain the existence of j as the result of someone initially producing a long English phoneme sequence e , according to $P(e)$, then transforming it into j , according to $P(j|e)$. The probability of our observed data $P(j)$ can be written as:

$$P(j) = \sum_e P(e) \cdot P(j|e)$$

We take $P(e)$ to be some fixed model of monolingual English phoneme production, represented as a weighted finite-state acceptor (WFSA). $P(j|e)$ is implemented as the initial, uniformly-weighted WFST C described in Section 2, with 15320 phonemic connections.

We next maximize $P(j)$ by manipulating the substitution table $P(j|e)$, aiming to produce a result such as shown in Figure 2. We accomplish this by composing the English phoneme model $P(e)$ WFSA with the $P(j|e)$ transducer. We then use the EM algorithm to train just the $P(j|e)$ parameters (inside the composition that predicts j), and guess the values for the individual phonemic substitutions that maximize the likelihood of the observed data $P(j)$.²

²In our experiments, we use the Carmel finite-state transducer package (Graehl, 1997), a toolkit with an algorithm for EM training of weighted finite-state transducers.

We allow EM to run until the $P(j)$ likelihood ratio between subsequent training iterations reaches 0.9999, and we terminate early if 200 iterations are reached.

Finally, we decode our test set of U.S. Senator names. Following Knight et al (2006), we stretch out the $P(j|e)$ model probabilities after decipherment training and prior to decoding our test set, by cubing their values.

Decipherment under the conditions of transliteration is substantially more difficult than solving letter-substitution ciphers (Knight et al., 2006; Ravi and Knight, 2008; Ravi and Knight, 2009) or phoneme-substitution ciphers (Knight and Yamada, 1999). This is because the target table contains significant non-determinism, and because each symbol has multiple possible fertilities, which introduces uncertainty about the length of the target string.

4.1 Baseline $P(e)$ Model

Clearly, we can design $P(e)$ in a number of ways. We might expect that the more the system knows about English, the better it will be able to decipher the Japanese. Our baseline $P(e)$ is a 2-gram phoneme model trained on phoneme sequences from the CMU dictionary. The second row (2a) in Figure 4 shows results when we decipher with this fixed $P(e)$. This approach performs poorly and gets all the Senator names wrong.

4.2 Consonant Parity

When training under non-parallel conditions, we find that we would like to keep our WFST C model small, rather than instantiating a fully-connected model. In the supervised case, parallel training allows the trained model to retain only those connections which were observed from the data, and this helps eliminate many bad connections from the model. In the unsupervised case, there is no parallel data available to help us make the right choices.

We therefore use prior knowledge and place a *consonant-parity* constraint on the WFST C model. Prior to EM training, we throw out any mapping from the $P(j|e)$ substitution model that does not have the same number of English and Japanese consonant phonemes. This is a pattern that we observe across a range of transliteration tasks. Here are ex-

Phonemic Substitution Model		Name Transliteration Error	
		whole-name error	norm. edit distance
1	$e \rightarrow j = \{ 1\text{-to-1}, 1\text{-to-2} \}$ + EM aligned with parallel data	40	25.9
2a	$e \rightarrow j = \{ 1\text{-to-1}, 1\text{-to-2} \}$ + decipherment training with 2-gram English $P(e)$	100	100.0
2b	$e \rightarrow j = \{ 1\text{-to-1}, 1\text{-to-2} \}$ + decipherment training with 2-gram English $P(e)$ + consonant-parity	98	89.8
2c	$e \rightarrow j = \{ 1\text{-to-1}, 1\text{-to-2} \}$ + decipherment training with 3-gram English $P(e)$ + consonant-parity	94	73.6
2d	$e \rightarrow j = \{ 1\text{-to-1}, 1\text{-to-2} \}$ + decipherment training with a word-based English model + consonant-parity	77	57.2
2e	$e \rightarrow j = \{ 1\text{-to-1}, 1\text{-to-2} \}$ + decipherment training with a word-based English model + consonant-parity + initialize mappings having consonant matches with higher probability weights	73	54.2

Figure 4: Results on name transliteration obtained when using the phonemic substitution model trained under different scenarios—(1) parallel training data, (2a-e) using only monolingual resources.

amples of mappings where consonant parity is violated:

K => a N => e e
EH => s a EY => n

Modifying the WFST C in this way leads to better decipherment tables and slightly better results for the U.S. Senator task. Normalized edit distance drops from 100 to just under 90 (row 2b in Figure 4).

4.3 Better English Models

Row 2c in Figure 4 shows decipherment results when we move to a 3-gram English phoneme model for $P(e)$. We notice considerable improvements in accuracy. On the U.S. Senator task, normalized edit distance drops from 89.8 to 73.6, and whole-name error decreases from 98 to 94.

When we analyze the results from deciphering with a 3-gram $P(e)$ model, we find that many of the Japanese phoneme test sequences are decoded into English phoneme sequences (such as “IH K R IH N” and “AE G M AH N”) that are not valid words. This happens because the models we used for decipherment so far have no knowledge of what constitutes a globally valid English sequence. To help the phonemic substitution model learn this information automatically, we build a word-based $P(e)$ from English phoneme sequences in the CMU dictionary and use this model for decipherment train-

ing. The word-based model produces complete English phoneme sequences corresponding to 76,152 actual English words from the CMU dictionary. The English phoneme sequences are represented as paths through a WFSA, and all paths are weighted equally. We represent the word-based model in compact form, using determinization and minimization techniques applicable to weighted finite-state automata. This allows us to perform efficient EM training on the cascade of $P(e)$ and $P(j|e)$ models. Under this scheme, English phoneme sequences resulting from decipherment are always analyzable into actual words.

Row 2d in Figure 4 shows the results we obtain when training our WFST C with a word-based English phoneme model. Using the word-based model produces the best result so far on the phonemic substitution task with non-parallel data. On the U.S. Senator task, word-based decipherment outperforms the other methods by a large margin. It gets 23 out of 100 Senator names exactly right, with a much lower normalized edit distance (57.2). We have managed to achieve this performance using only monolingual data. This also puts us within reach of the parallel-trained system’s performance (40% whole-name errors, and 25.9 word edit distance error) without using a single English/Japanese pair for training.

To summarize, the quality of the English phoneme

e	j	$P(j e)$	e	j	$P(j e)$	e	j	$P(j e)$	e	j	$P(j e)$	e	j	$P(j e)$	e	j	$P(j e)$	e	j	$P(j e)$			
AA	a	0.37	AY	a i	0.36	EH	e	0.37	HH	h	0.45	L	r	0.3	OY	a	0.27	SH	sh y	0.22	V	b	0.34
	o	0.25		o o	0.13		a	0.24		s	0.12		n	0.19		i	0.16		m	0.11		k	0.14
	i	0.15		e	0.12		o	0.12		k	0.09		ru	0.15		yu	0.1		r	0.1		m	0.13
	u	0.08		i	0.11		i	0.12		b	0.08		ri	0.04		oi	0.1		s	0.06		s	0.07
	e	0.07		a	0.11		u	0.06		m	0.07		t	0.03		ya	0.09		p	0.06		d	0.07
	oo	0.03		uu	0.05		oo	0.04		w	0.03		mu	0.02		yo	0.08		sa	0.05		r	0.04
	ya	0.01		yu	0.02		yu	0.01		p	0.03		m	0.02		e	0.08		h	0.05		t	0.03
	aa	0.01		u	0.02		ai	0.01		g	0.03		wa	0.01		o	0.06		b	0.05		h	0.02
				o	0.02					ky	0.02		ta	0.01		oo	0.02		t	0.04		sh	0.01
				ee	0.02					d	0.02		ra	0.01		ei	0.02		k	0.04		n	0.01
AE	a	0.52	B	b	0.41	ER	aa	0.47	IH	i	0.36	M	m	0.3	P	p	0.18	T	t	0.2	W	w	0.23
	i	0.19		p	0.12		a	0.17		e	0.25		n	0.08		pu	0.08		to	0.16		r	0.2
	e	0.11		k	0.09		u	0.08		a	0.15		k	0.08		n	0.05		ta	0.05		m	0.13
	o	0.08		m	0.07		o	0.07		u	0.09		r	0.07		k	0.05		n	0.04		s	0.08
	u	0.03		s	0.04		e	0.04		o	0.09		s	0.06		shi	0.04		ku	0.03		k	0.07
	uu	0.02		g	0.04		oo	0.03		oo	0.01		h	0.05		ku	0.04		k	0.03		h	0.06
	oo	0.02		t	0.03		ii	0.03					t	0.04		su	0.03		te	0.02		b	0.06
				z	0.02		yu	0.02		g	0.04		g	0.04		pa	0.03		s	0.02		t	0.04
				d	0.02		uu	0.02		b	0.04		b	0.04		t	0.02		r	0.02		p	0.04
				ch y	0.02		i	0.02		mu	0.03		mu	0.03		ma	0.02		gu	0.02		d	0.02
AH	a	0.31	CH	g	0.12	EY	ee	0.3	IY	i	0.25	N	n	0.56	PAUSE	pause	1.0	TH	k	0.21	Y	s	0.25
	o	0.23		k	0.11		a	0.22		ii	0.21		ru	0.09		pu	0.11		k	0.18			
	i	0.17		b	0.09		i	0.11		a	0.15		su	0.04		ku	0.1		m	0.07			
	e	0.12		sh	0.07		u	0.09		aa	0.12		mu	0.02		d	0.08		g	0.06			
	u	0.1		s	0.07		o	0.06		u	0.07		kk u	0.02		hu	0.07		p	0.05			
	ee	0.02		r	0.07		e	0.06		o	0.05		ku	0.02		su	0.05		b	0.05			
	oo	0.01		ch y	0.07		oo	0.05		oo	0.02		hu	0.02		bu	0.04		r	0.04			
	aa	0.01		p	0.06		ei	0.04		ia	0.02		to	0.01		ko	0.03		d	0.04			
				m	0.06		ii	0.02		ee	0.02		pp u	0.01		ga	0.03		u r	0.03			
				ch	0.06		uu	0.01		e	0.02		bi	0.01		sa	0.02		ny	0.03			
AO	o	0.29	D	d	0.16	F	h	0.18	JH	b	0.13	NG	tt o	0.21	R	r	0.53	UH	a	0.24	Z	to	0.14
	a	0.26		do	0.15		hu	0.14		k	0.1		ru	0.17		n	0.07		o	0.14		zu	0.11
	e	0.14		n	0.05		b	0.09		j y	0.1		n	0.14		ur	0.05		e	0.11		ru	0.11
	oo	0.12		to	0.03		sh i	0.07		s	0.08		kk u	0.1		ri	0.03		yu	0.1		su	0.1
	i	0.08		sh i	0.03		p	0.07		m	0.08		su	0.07		ru	0.02		ai	0.09		gu	0.09
	u	0.05		ku	0.03		m	0.06		t	0.07		mu	0.06		d	0.02		i	0.08		mu	0.07
	yu	0.03		k	0.03		r	0.04		j	0.07		dd o	0.04		t	0.01		uu	0.07		n	0.06
	ee	0.01		g u	0.03		s	0.03		h	0.07		tch i	0.03		s	0.01		oo	0.07		do	0.06
				b	0.03		ha	0.03		sh	0.06		pp u	0.03		m	0.01		aa	0.03		ji	0.02
				s	0.02		bu	0.02		d	0.05		ji	0.03		k	0.01		u	0.02		chi	0.02
AW	oo	0.2	DH	h	0.13	G	gu	0.13	K	k	0.17	OW	a	0.3	S	su	0.4	UW	u	0.39	ZH	m	0.17
	au	0.19		r	0.12		g	0.11		n	0.1		o	0.25		n	0.11		a	0.15		p	0.16
	a	0.18		b	0.09		ku	0.08		ku	0.1		oo	0.12		ru	0.05		o	0.13		t	0.15
	ai	0.11		w	0.08		bu	0.06		kk u	0.05		u	0.09		to	0.03		uu	0.12		h	0.13
	aa	0.11		t	0.07		k	0.04		to	0.03		i	0.07		ku	0.03		i	0.04		d	0.1
	e	0.05		p	0.07		b	0.04		su	0.03		ya	0.04		shi	0.02		yu	0.03		s	0.08
	o	0.04		g	0.06		to	0.03		sh i	0.02		e	0.04		ri	0.02		ii	0.03		b	0.07
	i	0.04		j y	0.05		t	0.03		r	0.02		uu	0.02		mu	0.02		e	0.03		r	0.05
	iy	0.02		d	0.05		ha	0.03		ko	0.02		ai	0.02		hu	0.02		oo	0.02		jj y	0.03
	ea	0.01		k	0.03		d	0.03		ka	0.02		ii	0.01		chi	0.02		ee	0.02		k	0.02

Figure 5: Phonemic substitution table learnt from non-parallel corpora. For each English phoneme, only the top ten mappings with $P(j|e) > 0.01$ are shown.

model used in decipherment training has a large effect on the learnt $P(j|e)$ phonemic substitution table (i.e., probabilities for the various phoneme mappings within the WFST C model), which in turn affects the quality of the back-transliterated English output produced when decoding Japanese.

Figure 5 shows the phonemic substitution table learnt using word-based decipherment. The mappings are reasonable, given the lack of parallel data. They are not entirely correct—for example, the mapping “S → s u” is there, but “S → s” is missing.

Sample end-to-end transliterations are illustrated in Figure 6. The figure shows how the transliteration results from non-parallel training improve steadily as we use stronger decipherment techniques. We note that in one case (LAUTENBERG), the decipherment mapping table leads to a correct answer

where the mapping table derived from parallel data does not. Because parallel data is limited, it may not contain all of the necessary mappings.

4.4 Size of Japanese Training Data

Monolingual corpora are more easily available than parallel corpora, so we can use increasing amounts of monolingual Japanese training data during decipherment training. The table below shows that using more Japanese training data produces better transliteration results when deciphering with the word-based English model.

Japanese training data (# of phoneme sequences)	Error on name transliteration task	
	whole-name error	normalized word edit distance
4,674	87	69.7
9,350	77	57.2

No Parallel Data Used.

Original	Correct Answer	Parallel Phonetic Training	Decipherment (Method 1)	Decipherment (Method 2)	Decipherment (Method 3)
スペンサー・ エーブラハム	SPENCER ABRAHAM	<u>SPENCER</u> <u>ABRAHAM</u>	<u>SPENCER</u> EDELMAN	SPACE CUOMO	<u>SPENCER</u> <u>ABRAHAM</u>
ダニエル・ アカカ	DANIEL AKAKA	<u>DANIEL</u> ACA KA	KOREA EAST	SCHERING EAST	DANIELLE ABACO
ウェイン・ アラード	WAYNE ALLARD	<u>WAYNE ALLARD</u>	CHOICE JOHN	<u>WAYNE</u> BYRD	<u>WAYNE ALLARD</u>
マックス・ ボーカス	MAX BAUCUS	<u>MAX BAUCUS</u>	MEESE JAMES	<u>MAX</u> BOOKS	<u>MAX</u> FOCUS
ボブ・ベネット	BOB BENNETT	<u>BOB BENNETT</u>	MY SCHERING	JACK BILLING	BOTH BENNING
ジョセフ・ バイデン	JOSEPH BIDEN	<u>JOSEPH BIDEN</u>	HOUSE LABOR	JAPAN <u>BIDEN</u>	<u>JOSEPH BIDEN</u>
ジェフ・ ビンガマン	JEFF BINGAMAN	<u>JEFF BINGAMAN</u>	JOHN PFEIFFER	<u>JEFF</u> BENJAMIN	<u>JEFF</u> BENJAMIN
フランク・ ローテンバーク	FRANK LAUTENBERG	<u>FRANK</u> LAUTENBACH	SUN FLINT	FRANCE <u>LAUTENBERG</u>	FRANCE <u>LAUTENBERG</u>

Figure 6: Results for end-to-end name transliteration. This figure shows the correct answer, the answer obtained by training mappings on parallel data (Knight and Graehl, 1998), and various answers obtained by deciphering non-parallel data. Method 1 uses a 2-gram $P(e)$, Method 2 uses a 3-gram $P(e)$, and Method 3 uses a word-based $P(e)$.

4.5 $P(j|e)$ Initialization

So far, the $P(j|e)$ connections within the WFST C model were initialized with uniform weights prior to EM training. It is a known fact that the EM algorithm does not necessarily find a global minimum for the given objective function. If the search space is bumpy and non-convex as is the case in our problem, EM can get stuck in any of the local minima depending on what weights were used to initialize the search. Different sets of initialization weights can lead to different convergence points during EM training, or in other words, depending on how the $P(j|e)$ probabilities are initialized, the final $P(j|e)$ substitution table learnt by EM can vary.

We can use some prior knowledge to initialize the probability weights in our WFST C model, so as to give EM a good starting point to work with. Instead of using uniform weights, in the $P(j|e)$ model we set higher weights for the mappings where English and Japanese sounds share common consonant phonemes.

For example, mappings such as:

$$\begin{array}{ll} N \Rightarrow n & N \Rightarrow a \ n \\ D \Rightarrow d & D \Rightarrow d \ o \end{array}$$

are weighted X (a constant) times higher than other mappings such as:

$$\begin{array}{ll} N \Rightarrow b & N \Rightarrow r \\ D \Rightarrow B & EY \Rightarrow a \ a \end{array}$$

in the $P(j|e)$ model. In our experiments, we set the value X to 100.

Initializing the WFST C in this way results in EM learning better substitution tables and yields slightly better results for the Senator task. Normalized edit distance drops from 57.2 to 54.2, and the whole-name error is also reduced from 77% to 73% (row 2e in Figure 4).

4.6 Size of English Training Data

We saw earlier (in Section 4.4) that using more monolingual Japanese training data yields improvements in decipherment results. Similarly, we hypothesize that using more monolingual English data can drive the decipherment towards better transliteration results. On the English side, we build different word-based $P(e)$ models, each trained on different amounts of data (English phoneme sequences from the CMU dictionary). The table below shows that deciphering with a word-based English model

built from more data produces better transliteration results.

English training data (# of phoneme sequences)	Error on name transliteration task	
	whole-name error	normalized word edit distance
76,152	73	54.2
97,912	66	49.3

This yields the best transliteration results on the Senator task with non-parallel data, getting 34 out of 100 Senator names exactly right.

4.7 Re-ranking Results Using the Web

It is possible to improve our results on the U.S. Senator task further using external monolingual resources. Web counts are frequently used to automatically re-rank candidate lists for various NLP tasks (Al-Onaizan and Knight, 2002). We extract the top 10 English candidates produced by our word-based decipherment method for each Japanese test name. Using a search engine, we query the entire English name (first and last name) corresponding to each candidate, and collect search result counts. We then re-rank the candidates using the collected Web counts and pick the most frequent candidate as our choice.

For example, *France Murkowski* gets only 1 hit on Google, whereas *Frank Murkowski* gets 135,000 hits. Re-ranking the results in this manner lowers the whole-name error on the Senator task from 66% to 61%, and also lowers the normalized edit distance from 49.3 to 48.8. However, we do note that re-ranking using Web counts produces similar improvements in the case of parallel training as well and lowers the whole-name error from 40% to 24%.

So, the re-ranking idea, which is simple and requires only monolingual resources, seems like a nice strategy to apply at the end of transliteration experiments (during decoding), and can result in further gains on the final transliteration performance.

5 Comparable versus Non-Parallel Corpora

We also present decipherment results when using comparable corpora for training the WFST C model. We use English and Japanese phoneme sequences derived from a parallel corpus containing 2,683 phoneme sequence pairs to construct comparable corpora (such that for each Japanese phoneme se-

quence, the correct back-transliterated phoneme sequence is present somewhere in the English data) and apply the same decipherment strategy using a word-based English model. The table below compares the transliteration results for the U.S. Senator task, when using comparable versus non-parallel data for decipherment training. While training on comparable corpora does have benefits and reduces the whole-name error to 59% on the Senator task, it is encouraging to see that our best decipherment results using only non-parallel data comes close (66% error).

English/Japanese Corpora (# of phoneme sequences)	Error on name transliteration task	
	whole-name error	normalized word edit distance
Comparable Corpora (English = 2,608 Japanese = 2,455)	59	41.8
Non-Parallel Corpora (English = 98,000 Japanese = 9,350)	66	49.3

6 Conclusion

We have presented a method for attacking machine transliteration problems without parallel data. We developed phonemic substitution tables trained using only monolingual resources and demonstrated their performance in an end-to-end name transliteration task. We showed that consistent improvements in transliteration performance are possible with the use of strong decipherment techniques, and our best system achieves significant improvements over the baseline system. In future work, we would like to develop more powerful decipherment models and techniques, and we would like to harness the information available from a wide variety of monolingual resources, and use it to further narrow the gap between parallel-trained and non-parallel-trained approaches.

7 Acknowledgements

This research was supported by the Defense Advanced Research Projects Agency under SRI International's prime Contract Number NBCHD040058.

References

- Y. Al-Onaizan and K. Knight. 2002. Translating named entities using monolingual and bilingual resources. In *Proc. of ACL*.
- 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*, 39(4):1–38.
- Y. Goldberg and M. Elhadad. 2008. Identification of transliterated foreign words in Hebrew script. In *Proc. of CILing*.
- D. Goldwasser and D. Roth. 2008a. Active sample selection for named entity transliteration. In *Proc. of ACL/HLT Short Papers*.
- D. Goldwasser and D. Roth. 2008b. Transliteration as constrained optimization. In *Proc. of EMNLP*.
- S. Goldwater and L. Griffiths, T. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proc. of ACL*.
- J. Graehl. 1997. Carmel finite-state toolkit. <http://www.isi.edu/licensed-sw/carmel>.
- L. Haizhou, Z. Min, and S. Jian. 2004. A joint source-channel model for machine transliteration. In *Proc. of ACL*.
- U. Hermjakob, K. Knight, and H. Daume. 2008. Name translation in statistical machine translation—learning when to transliterate. In *Proc. of ACL/HLT*.
- F. Huang, S. Vogel, and A. Waibel. 2004. Improving named entity translation combining phonetic and semantic similarities. In *Proc. of HLT/NAACL*.
- S. Karimi, F. Scholer, and A. Turpin. 2007. Collapsed consonant and vowel models: New approaches for English-Persian transliteration and back-transliteration. In *Proc. of ACL*.
- A. Klementiev and D. Roth. 2008. Named entity transliteration and discovery in multilingual corpora. In *Learning Machine Translation*. MIT press.
- K. Knight and J. Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4):599–612.
- K. Knight and K. Yamada. 1999. A computational approach to deciphering unknown scripts. In *Proc. of the ACL Workshop on Unsupervised Learning in Natural Language Processing*.
- K. Knight, A. Nair, N. Rathod, and K. Yamada. 2006. Unsupervised analysis for decipherment problems. In *Proc. of COLING/ACL*.
- J. Kuo, H. Li, and Y. Yang. 2006. Learning transliteration lexicons from the web. In *Proc. of ACL/COLING*.
- H. Li, C. Sim, K., J. Kuo, and M. Dong. 2007. Semantic transliteration of personal names. In *Proc. of ACL*.
- M. Nagata, T. Saito, and K. Suzuki. 2001. Using the web as a bilingual dictionary. In *Proc. of the ACL Workshop on Data-driven Methods in Machine Translation*.
- J. Oh and H. Isahara. 2006. Mining the web for transliteration lexicons: Joint-validation approach. In *Proc. of the IEEE/WIC/ACM International Conference on Web Intelligence*.
- S. Ravi and K. Knight. 2008. Attacking decipherment problems optimally with low-order n-gram models. In *Proc. of EMNLP*.
- S. Ravi and K. Knight. 2009. Probabilistic methods for a Japanese syllable cipher. In *Proc. of the International Conference on the Computer Processing of Oriental Languages (ICCPOL)*.
- T. Sherif and G. Kondrak. 2007a. Bootstrapping a stochastic transducer for arabic-english transliteration extraction. In *Proc. of ACL*.
- T. Sherif and G. Kondrak. 2007b. Substring-based transliteration. In *Proc. of ACL*.
- R. Sproat, T. Tao, and C. Zhai. 2006. Named entity transliteration with comparable corpora. In *Proc. of ACL*.
- T. Tao, S. Yoon, A. Fister, R. Sproat, and C. Zhai. 2006. Unsupervised named entity transliteration using temporal and phonetic correlation. In *Proc. of EMNLP*.
- J. Wu and S. Chang, J. 2007. Learning to find English to Chinese transliterations on the web. In *Proc. of EMNLP/CoNLL*.
- D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proc. of ACL*.
- S. Yoon, K. Kim, and R. Sproat. 2007. Multilingual transliteration using feature based phonetic method. In *Proc. of ACL*.
- D. Zelenko and C. Aone. 2006. Discriminative methods for transliteration. In *Proc. of EMNLP*.