

---

# Machine Transliteration (Knight & Graehl, ACL 97)

Kevin Duh

UW Machine Translation Reading  
Group, 11/30/2005

# Transliteration & Back-transliteration

---

- Transliteration:
  - Translating proper names, technical terms, etc. based on phonetic equivalents
  - Complicated for language pairs with different alphabets & sound inventories
  - E.g. “computer” --> “konpyuutaa” コンピューター
- Back-transliteration
  - E.g. “konpyuuta” --> “computer”
  - Inversion of a lossy process

# Japanese/English Examples

---

- Some notes about Japanese:
  - Katakana phonetic system for foreign names/loan words
  - Syllabary writing:
    - e.g. one symbol for “ga” ガ, one for “gi” ギ
  - Consonant-vowel (CV) structure
  - Less distinction of L/R and H/F sounds
- Examples:
  - Golfbag --> goruhubaggu ゴルフバッグ
  - New York Times --> nyuuyooku taimuzu ニューヨーク タイムズ
  - Ice cream --> aisukuriimu アイスクリーム

# The Challenge of Machine Back-transliteration

---

- Back-transliteration is an important component for MT systems
  - For J/E: Katakana phrases are the largest source of phrases that do not appear in bilingual dictionary or training corpora
- Claims:
  - Back-transliteration is less forgiving than transliteration
  - Back-transliteration is harder than romanization
  - For J/E, not all katakana phrases can be “sounded out” by back-transliteration
    - word processing --> waapuro
    - personal computer --> pasokon

# Modular WSA and WFSTs

---

- $P(w)$  - generates English words
  - $P(e|w)$  - English words to English pronunciation
  - $P(j|e)$  - English to Japanese sound conversion
  - $P(k|j)$  - Japanese sound to katakana
  - $P(o|k)$  - katakana to OCR
- 
- Given a katana string observed by OCR, find the English word sequence  $w$  that maximizes

$$\sum_e \sum_j \sum_k P(w)P(e|w)P(j|e)P(k|j)P(o|k)$$

---

# Two Potential Solutions

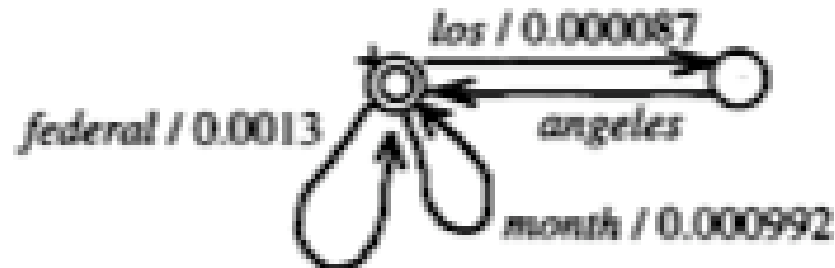
---

- Learn from bilingual dictionaries, then generalize
  - Pro: Simple supervised learning problem
  - Con: finding direct correspondence between English alphabets and Japanese katakana may be too tenuous
- Build a generative model of transliteration, then invert (Knight & Graehl's approach):
  1. An English phrase is written
  2. A translator pronounces it in English
  3. The pronunciation is modified to fit the Japanese sound inventory
  4. The sound is converted to katakana
  5. Katakana is written

# Word Sequence WSA: $P(w)$

---

- Get word sequence probabilities from a 262k list of words/phrases from WSJ + online English name list + online gazeteer of place names:

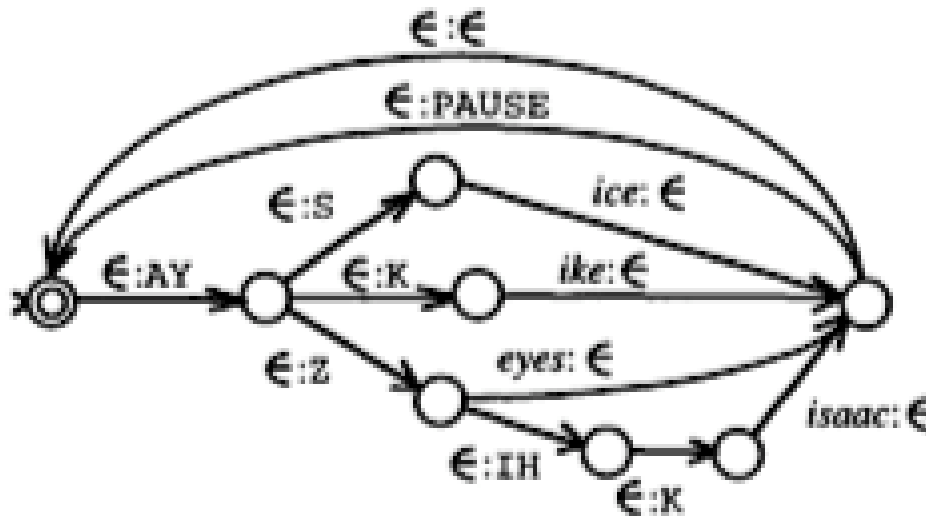


- Removed high-frequency words (e.g. function words, auxiliary verbs) that are not usually transliterated
  - Built separate WSA for person names
-

# Word to English Sound WFST: $P(e|w)$

---

- Use CMU Pronunciation Dictionary to generate a phoneme-tree based WFST:
  - 50k words, 25 consonants, 14 vowels, 1 PAUSE



- Alternative solution: general letter-to-sound FST

# English Sound to Japanese Sound WFST: P(j|e) [1/2]

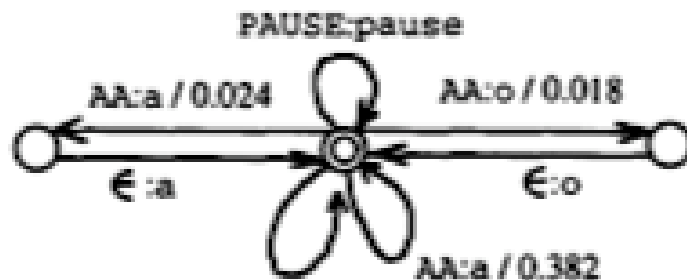
---

- What is the target Japanese sound inventory?
  - Option1: Katakana sounds (e.g. English sound K maps to one of カ ka, キ ki, ク ku, ケ ke, コ ko)
    - (P R OW PAUSE S AA K ER) --> (pu ro pause sa kka) プロサッカー
    - Con: does not generalize: English K --> Japanese k sound
  - Option2: New inventory (Knight/Graehl approach)--
    - 5 vowels, 33 consonants
    - (P R OW PAUSE S AA K ER) --> (p u r o pause s a kk a a)
    - Note: long Japanese vowels are written as two symbols (a a)

# English Sound to Japanese Sound WFST: $P(j|e)$ [2/2]

---

- WSFT is learned from a English-Katakana dictionary (8k pairs)
  - Generate sound sequences from pronunciation models  $P(e|w)$  &  $P(j|k)$
  - Symbol mapping probabilities trained by EM
  - Build WSFT from the probabilities:



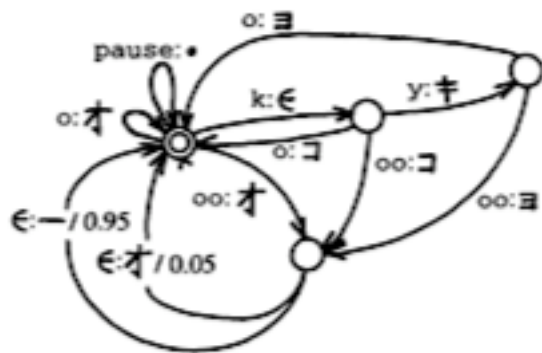
- (More details in the paper)

# Japanese sound to Katakana

## WFST: P(k|j)

---

- 2 stage WFST:
  - Stage1: merges long Japanese vowel (a a) to one symbol (aa)
    - idea is to consume whole symbol before producing katakana
  - Stage2: conversion to Katakana
    - Some art/skill required here: based on corpus analysis and guidelines from a Japanese textbook



# Katakana to OCR: $p(o|k)$

---

- Models OCR confusions:

k	o	$P(o k)$
エ	エ	0.492
	ヱ	0.434
	エ	0.042
	7	0.011
ヱ	ヱ	1.000
ハ	ハ	0.964
	ノ	0.036

- Probabilities generated by:
  1. Begin with katakana words (19,500 characters)
  2. Print them out and apply OCR
  3. Align with EM training

# Example to tie it all together

---

- Input from OCR: マスクーズトーチメント
- Convert string to 12state/11arc FSA
- Compose with  $p(k|o)$  --> 12state/15arc WFSA
- Compose with  $p(j|k)$  --> 28state/31arc WFSA
  - Max score sequence: m a s u t a a z u t o o c h i m e n t o
- Compose with  $p(e|j)$  --> 62state/241arc WFSA
  - Max score sequence: M A E S T A E A E D H U H T A O A O C H I H M  
E H N T A O
- Compose with  $p(w|e)$  --> 2982state/4601arc WFSA
  - Max score sequence: masters tone am ent awe
- Rescore with  $p(w)$ 
  - Max score sequence: masters tournament

# Experiments

---

- Task: Back-transliterate the katakana of names of 100 U.S. politicians
  - E.g. jyon.buro, maiku.dewain, aruhonsu.damatto
- Compare machine with 4 humans:
  - Humans are native English speakers who are news aware.
  - However, they are not experts in Japanese phonetics

	Human	Machine
Correct	27%	64%
Phonetically equivalent but misspelled	7%	12%
Incorrect	66%	24%

---

# Discussions

---

- Back-transliteration for other languages (e.g. Arabic/Chinese to English)
- Incorporation into MT and Noun-entity tagger?
- What is the bottleneck/challenge of back-transliteration?
- WFST approach vs. other approaches?