Phoneme-based Statistical Transliteration of Foreign Names for OOV Problem

GAO Wei

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Systems Engineering and Engineering Management

Supervised by

Prof. WONG Kam-Fai and Prof. LAM Wai

Abstract

Given a source language term, machine transliteration is to automatically generate the phonetic equivalents in a target language. It is useful in many cross language applications. Recently, there are increasing concerns about automatic transliteration, especially with languages with significant distinctions in their phonetic representations, e.g. English and Chinese. Despite many cross-language applications in English/Chinese, machine transliteration between the two languages has not been studied comprehensively.

Existing English-Chinese transliteration techniques are typically based on source-channel framework, e.g. IBM SMT model. The accuracy of this model is rather low. In this thesis, we propose to use a direct approach for English-to-Chinese transliteration. We propose two direct transliteration models: In the first model, we model the problem as direct phonetic mapping from English phonemes to a set of rudimentary Chinese phonetic symbols plus dynamically discovered mapping units from training process. An effective algorithm for alignment of phoneme chunks is presented. In the second model, contextual features of each phoneme are taken into consideration by means of Maximum Entropy formalism, and it is further refined with the precise alignment scheme based on phoneme chunks. We compared the direct approaches with the source-channel baseline implemented with the IBM SMT model, and showed that the second approach is significantly superior.

摘要

机器音译就是根据发音将给定的源语言中的专有名词自动翻译成目标语言中对应的词汇。它在跨语言应用中有较多的用途。近来，自动音译的研究受到越来越多的关注，特别是当音译所涉及的两种语言在声音的表示方面有很大差别的情况，比如英文和中文。尽管关于中英文的跨语言应用研究有很多，但是这两种语言之间的机器音译还没有经过全面广泛的探索。

典型的英中文音译技术是基于信–源模型，例如，采用IBM统计机器翻译模型。这个模型应用在机器音译上准确率非常低。在本论文中，我们提出了采用从英文到中文直接音译的方法。我们运用了两种直接音译模型：在第一个模型里，音译问题被建模成从英文音素集合到一个基本中文音素集合的直接映射，再附以从训练过程中动态探测得到的发音映射单元，并且我们展示了一个用于对齐音素群的有效算法。在第二个模型中，我们通过采用最大熵理论考虑了每个音素所在的上下文特征，并且近一步使用音素群精确对齐的方法改善这个模型。我们比较了直接音译方法和采用IBM统计机器翻译技术实现的信–源模型系统，结果显示我们的第二个方法有显著的优势。

Acknowledgement

I would like to thank my supervisor, Prof. Kam-Fai Wong. He gives me not only full freedom to choose my research topic, but also a lot of guidance throughout the path, along which I have transferred from an engineer to a research-oriented academic. He constantly devotes great enthusiasms to this research topic and provides me with many helpful supports. Although being extremely occupied by his daily schedule, he often has to take much time and a lot pain to improve my paper drafts due to my poor English presentation. I learn plenty of useful things from his comments, revisions and discussions, which can teach me how to write research papers. Also many thanks to Prof. Wong for his continuous encouragements on my research progress as well as participation in academic activities.

Special thanks go to my co-supervisor, Prof. Wai Lam, from whom I can readily seek help for my confusions on critical questions concerning mathematical modeling of machine transliteration. Prof. Lam is an outstanding researcher on machine learning, information retrieval and natural language processing. His high-level remarks and enlightenments via our discussions help me a lot for correcting the way I conduct this research.

I should also appreciate the great help from Prof. Helen Meng, who provides me with the valuable corpus from LDC. Without her generous help, the comprehensive experimentations on this research would be impossible.

I am grateful to several active researchers at the Information Science Institute of the University of Southern California: Prof. Kevin Knight, Prof. Daniel Marcu and Mr. Ulrich Germann. The online discussions about SMT model with Prof. Knight give me much insight to the methodology for use. He is always ready for my questions. His quick replies and generosity on exchange of ideas are much inspiring to me. Prof. Marcu and Mr. Germann also provide useful supports via emails for the use of ISI MT toolkits even though they are superbly occupied persons.

Finally, I'd like to thank my parents for their deepest concerns for my study and life in Hong Kong. Dongmei always acts as my patient listener and encouraging supporter. I am deeply indebt to their love.

To my loving parents and future wife.

Bibliographic Notes

Portions of the thesis have appeared or will appear elsewhere.

Chapter 4 is based on the paper "Phoneme-based transliteration of foreign names for OOV problem" co-authored with Kam-Fai Wong and Wai Lam, which has been appeared in the proceedings of the First International Joint Conference on Natural Language Processing (IJCNLP-04). And the revised contribution will also appear on the special edition for IJCNLP-04 by the book of Lecture Notes on Artificial Intelligence, Springer Verlag Publication.

Chapter 5 is based on the paper "Improving transliteration of foreign names by precise alignment of phoneme chunks and using contextual features" co-authored with Kam-Fai Wong and Wai Lam, which was submitted to the First Asian Information Retrieval Symposium (AIRS-04).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 What is Transliteration?

The adoption of a foreign name into one language is usually a process of adjusting its original pronunciation to suit the phonological regularities in the target language. This procedure of phonetically "translating" foreign names is called *transliteration*. For instance, English name "Britain" is transliterated into Mandarin Chinese as " 不列颠 " (/bu lie dian/[1]), which undergoes significant transformations to accommodate the phonological characteristics of Chinese language.

The growing trend of globalization demands effective and efficient worldwide information access across language barriers. Automatic name translation mechanisms are recognized as important issues in many cross-language applications. Cross-lingual information retrieval (CLIR) involves keyword translation from the source to target language and document translation in the opposite direction. Proper names are especially frequent targets in queries. Similar demands are also becoming imminent in machine translation (MT) and spoken language processing, such as cross-lingual spoken document retrieval and spoken language translation. Contemporary lexicon-based translation techniques are ineffective as translation dictionaries can never be comprehensive for proper names. New names appear almost daily and they become unregistered vocabulary in the lexicon. This is known as the *Out-Of-Vocabulary* (OOV) problem in lexicography [16]. The lack of translation for OOV names hinders the performance of the underlying applications. One way to solve this problem is not to rely on a dictionary alone but to combine it with *machine transliteration* techniques [19].

[1]Mandarin pinyin is used as phonetic representations of Chinese characters throughout this paper. For simplicity, we ignore the four tones in the pinyin system.

Machine transliteration is classified in two directions: *forward* and *backward*. Given a name pair *(o, t)*, where *o* is the original in one language and *t* is the transliterated name of *o* in another language, forward transliteration (or transliteration) is converting from *o* to *t*; and backward transliteration (or back-transliteration) is retrieving the correct *o* given *t*. For example, "Britain→ 不列颠 " is a typical forward transliteration pair and the corresponding back-transliteration is " 不列颠 →Britain".

Regardless of the directions, machine transliteration can also be classified in terms of the level of units being transliterated, i.e. *phoneme-based* and *grapheme-based*, which are sometimes referred to as the *pivot* and *direct*[2] methods [21], respectively. Phoneme-based transliteration is done in steps: (a). convert the words into pronunciation symbols, i.e. phonemes; (b). transliterate the phonemes of the source language into the counterparts in the target language; (c). convert the resultant phonemes to the target words. In grapheme-based method, source words are transcribed to the target words based on graphematic units directly without making use of their phonetic representations.

In our research, we concentrate on phoneme-based techniques for automatic transliteration of foreign names in English to their Chinese counterparts, i.e. forward transliteration.

1.2 Existing Problems

Unlike other research areas in human language technologies, machine transliteration is relatively immature. It suffers from the following problems:

1. **Ambiguous Standards:** In practice, transliterations are hand-coded using rules of thumb. Existing rule bases are compiled manually. They are not easy to expand and are mostly subjective, i.e. they are subjected to the interpretation of individual producers. *De facto* standards have been established, but the rules are often inconsistently used. Dialectical discrepancies may further aggravate the inconsistency of transliteration standards. Table 1.1 lists some of these examples appearing in the three major Chinese language communities: mainland China, Hong Kong and Taiwan[3]. Due to dialectical differences, each region uses its respective set of rules for transliteration. However, we

[2]Note that the *direct method* here refers to transliteration based on graphemes directly instead of involving phoneme-level representations. It is not the *direct transliteration model* we will propose in this thesis. See later chapters for details.

[3]The regional transliterations and their usage shown in the table were extracted from the major news websites of the three regions using a search engine.

Original Name	Chinese Transliterations		Mainland	Hong Kong	Taiwan
Al Qaeda	艾塔	/ai ta/			√
	埃塔	/ai ta/	√	√	
	阿盖达	/a gai da/	√	√	
	盖达	/gai da/	√	√	√
(Mohammed) Atta	艾塔	/ai ta/	√		√
	阿塔	/a ta/	√	√	√
Bin Laden	本拉登	/ben la deng/	√	√	√
	宾拉登	/bin la deng/	√	√	√
	本拉丹	/ben la dan/	√	√	
	宾拉丹	/bin la dan/			√
Enron	安然	/an ran/	√	√	√
	安龙	/an long/	√	√	√
	安隆	/an long/	√	√	√
	恩龙	/en long/	√		√
	恩隆	/en long/			√
Hussein	侯塞因	/hou sai yin/	√	√	
	海珊	/hai shan/			√
	哈珊	/ha shan/			√
Inter(net)	因特(网)	/yin te (wang)/	√	√	√
	英特(网)	/ying te (wang)/	√	√	√
SARS	萨斯	/sa si/	√	√	√
	沙士	/sha shi/	√	√	√
	沙斯	/sha si/	√		√

Table 1.1: Some transliteration confusions found in Chinese websites and news media.

observe that multiple transliterations of the same original name coexist even in the same region. Moreover, transliteration initially produced by one region could sometimes spread over the entire communities. The transliterated word would then be assimilated as the equivalents of the local transliteration in other regions. Thus, the established rules and standards actually have been undermined by the existent transliteration discrepancies.

2. **Specific to Applications:** Several methodologies for English-Chinese transliteration have been proposed for specific applications, such as information retrieval [8, 22, 23, 33], acquisition or extraction of equivalent word pairs from parallel corpus [14, 20], and construction of named entity translation dictionary [13, 34]. Since they are specific to desig-

nated applications, comprehensive experiments evaluating transliteration accuracy, e.g. comparing machine-generated transliteration with the standard counterparts, are usually missing. Beside [23, 33], which present syllable errors based on *edit distance*, others only evaluate the performance of the underlying application, e.g. precision/recall in information retrieval. Thus, true transliteration accuracy practically remains unknown.

3. **Lack of Comparisons:** There is neither a widely recognized benchmark nor consistent measurement for machine transliteration, making it hard for comparative evaluation. Furthermore, the evaluation process often requires human judgement. The accuracies of existing transliteration techniques are unsatisfactory. But there is no useful guidelines to improve them.

1.3 Objectives

Our method is phoneme-based. Grapheme-Phoneme transformation and Pinyin-to-Hanzi[4] conversion applied in the phoneme-based methods are extensively studied in other areas like *Text-To-Speech* (TTS) in speech synthesis and *Speech-To-Text* in speech recognition. In our research, we focus on the intermediate processes for transliterating phoneme pairs. Addressing the aforementioned problems, in this thesis, we propose to:

- Investigate existing approaches for English-Chinese transliteration tasks and identify their pitfalls;
- Identify the central problems of phonetic transliteration between English and Chinese phoneme sequences.
- Apply the well-known methodologies as the baseline and propose improved models to overcome the problems identified.
- Compare the transliteration performance between the proposed methods and the baseline model by experimentations.

1.4 Outline

In this introductory chapter, we have given an overview and definition of the machine transliteration problem.

[4]It is also referred to as the process converting generalized initial and final (GIF) symbols to Chinese characters.

Chapter 2 presents a survey on the existing transliteration methodologies. We will highlight the related contributions for English-Chinese transliteration. Our stress will be given to the well-known *source-channel* model and an English-to-Chinese transliteration system based on it [33].

Chapter 3 addresses the limitations of the source-channel model for English-to-Chinese transliteration problem. The phoneme transliteration module described in [33] is implemented using IBM Statistical Machine Translation (SMT) toolkits as the baseline for the comparative experiments with our proposed approaches in later chapters. The preliminary experimental results on this baseline is presented.

Chapter 4 proposes a *direct transliteration model* **Direct-1** to partly overcome the problems faced by the source-channel model. The experiments on the **Direct-1** model are conducted, which include but are not limited to the comparisons with the baseline system.

Chapter 5 proposes an *improved* direct transliteration model, which is referred to as **Direct-2**, by incorporating flexible contextual features of neighboring phoneme dependencies. Then the model is further refined by **Direct-2R** using a precise alignment scheme. Comparative experiments on the **Direct-2** and the **Direct-2R** are conducted. Their improvements on performance are justified compared to the baseline.

Finally, Chapter 6 summarizes the thesis and discusses future extensions and applications of the proposed approaches.

□ **End of chapter.**

Chapter 2

Background

Several approaches have been proposed for automatic name transliteration between various language pairs in both directions. Regardless of languages and directions, the underlying mathematical models are more or less based on statistics, where a number of previous works typically employed source-channel framework as their probabilistic foundation. In the English-Chinese arena, although approaches varied with applications employing transliteration, statistical strategies also dominated other methods. This chapter will first retrospect previous works based on source-channel model, and then investigate the major contributions in English-Chinese arena, in which we will highlight statistical approaches. Note that there are various other approaches proposed for transliterating English to Asian languages, such as Korean and Japanese. Because they were implemented and tested specifically for the targeted language pairs, it would be hard to directly compare these systems with that for English and Chinese. Thus, we focus our research on comparisons of methods dealing with our designated languages.

2.1 Source-channel Model

Based on the *Bayes' theorem*, [19] described a generative model, in which they adopted finite state transducers and a decoder to transform transliterated names in Japanese *katakana* back to their origins in English. Given a katakana string o observed by an optical character recognition (OCR) process, the system aimed to find the English word w that maximizes $P(w|o)$ [19]:

$$\operatorname*{argmax}_{w} P(w|o) = \operatorname*{argmax}_{w} \{P(w) \cdot P(e|w) \cdot P(j|e) \cdot P(k|j) \cdot P(o|k)\} \tag{2.1}$$

where the five probability distributions denote:

$P(w)$— the probability of the generated written English word sequence w;

$P(e|w)$— the probability of pronounced English word sequence w based on English sound e;

$P(j|e)$— the probability of converted English sound units e based on Japanese sound units j;

$P(k|j)$— the probability of the Japanese sound units j based on the katakana writing k;

$P(o|k)$— the probability of katakana writing k based on the observed OCR pattern o.

The individual models were implemented in a *weighted finite state acceptor* (WFSA) for $P(w)$ and the four *weighted finite state transducers* (WFST) for the other four distributions to map corresponding source sequences to the targets. The five automatas were created using different data sources: $P(w)$ was estimated adopting a simple unigram scoring method by using a 262,000-entry frequency list; $P(e|w)$ was built from the on-line CMU pronunciation dictionary without stress marks; $P(j|e)$ was approximated using *EM* algorithm from 8,000 pairs of English-Japanese sound sequences; $P(k|j)$ was constructed manually according to Japanese sound to katakana rules; and finally, $P(o|k)$ was approximated by estimating the symbol-mapping probabilities between the katakana symbols in the 19,500 original katakana words and the OCR-generated katakana symbols from a printout of those words. Using a general composition algorithm, an integrated model combining the five separate automatas was formed. They implemented *k-shortest-paths* algorithms for extracting the best transliterations from the large resulting WFSA given a source katakana word.

In general, the generative model above can be simplified as a maximization problem without intermediate steps, assuming that the parameters could be reliably obtained from the given source and target name pairs:

$$\underset{T}{\operatorname{argmax}} \Pr(T|S) = \underset{T}{\operatorname{argmax}} \{\Pr(S|T) \times \Pr(T)\} \tag{2.2}$$

where we use Pr(.) to denote a general probability distribution, and S and T are the source and target names, respectively. This fundamental equation is also referred to as *source-channel* model.

[19]'s method is the first attempt in applying statistics to the transliteration problem. Some steps in the model have to be carried out by hand. Errors could propagate between generations. Enlightened by their initiatives, the source-channel model was extended for different tasks: [32] applied

source-channel based automatas to Arabic-to-English back-transliteration. [1] used the same model to combine phoneme-based and grapheme-based approaches for Arabic-to-English transliteration. A similar model was proposed by [5] to combine the phoneme and grapheme sources for Japanese-to-English back-transliteration. It was also applied to English-Korean language pairs: English-to-Korean transliteration by [21, 28] and Korean-to-English back-transliteration by [15] used pure statistical estimation rather than partially hand-coded generative automatas.

2.2 Transliteration for English-Chinese

Previous models for English-Chinese transliteration/back-transliteration vary with the application domains.

2.2.1 Rule-based Approach

[34] was the first contribution developing a grapheme-based transliteration mechanism from English proper nouns to Chinese for a multilingual text generation system. In this approach, they used handcrafted rules. The algorithm first syllabified English words based on a rule set and instance learning. The syllabification module identified syllable boundaries based on consonant clusters and vowels. A sub-syllabification procedure then divided each identified syllable into the form of consonant-vowel, i.e. conforming to the mono-syllabic nature of Chinese. The sub-syllables are then mapped to the pinyin equivalents and consequently to the Chinese characters by means of two handcrafted mapping tables. This approach is intensively ad-hoc and dialect dependant. No report on its performance was provided.

2.2.2 Similarity-based Framework

[8, 22] proposed a similarity-based method for CLIR to find the best matching words from a set of target candidates in English given a Chinese query containing foreign proper names. Similarity-based approaches were tested in the grapheme level as well as phoneme level. Furthermore, [22] addressed the problem of ad hoc assigned phonetic similarities by developing a learning algorithm to automatically acquire the similarities from a training corpus. With the learning algorithm, the labor of assigning phonetic similarities between two languages could be removed leading to improved transliteration performance. The accuracy was measured by the average rankings of the correct candidates found. Compared with the generative model [2, 19, 32] for other language pairs, the similarity-based framework directly addressed

the problem of similarity measurements, and could be evaluated without human judgment [22]. However, this approach assumed that a set of target candidates could be identified first and could only find the best matching words from the candidate set. The mechanism would not work if the correct target was not included in the candidate set. Thus, its application is limited.

2.2.3 Direct Semi-Statistical Approach

[23] presented a learning algorithm for transliteration of OOV names from English to Chinese in the context of Cross-Language Spoken Document Retrieval. For clarity, the process is illustrated by Figure 2.1, which is excerpted from [23].

They first used a set of handcrafted phonological rules by adding or dropping proper phonemes to normalize English syllables into consonant-vowel format. This aimed to overcome some of the phonological differences between the two languages. The process of cross-lingual phonetic mapping (CLPM) then applied a set of automatically generated one-to-one phonetic transformation rules to map English phonemes to their Chinese counterparts. These rules were learned from aligned parallel data using *transformation-based error-driven learning*(TEL) [6]. The pinyin syllabic constraints were then added to the Chinese phoneme sequences generated by CLPM for eliminating the errors in the sequences. A phoneme lattice of pinyin sub-syllables were generated based on a confusion matrix obtained from the mapping differences between reference phonemes and output phonemes. They searched the phoneme lattice exhaustively for Chinese phonetic sequences that could constitute legitimate syllables to create a syllable graph. Finally, a syllable bigram language model was applied together with the probabilities derived from the confusion matrix to search the graph to find the most probable syllable sequence. The transliteration performance measured by the syllable accuracy of was 47.5%.

One shortcoming of this method is that the manually enumerated rules initiated for CLPM are unable to balance all the phonological discrepancies of two names. This may introduce errors to probability estimation in later stages. For example, the following rules:

`[rule 1] Insert a reduced nuclei /ax/ between clustered consonants.`
`Clinton`→`/`*K L* `IH N T AH N/`→`/K` *ax* `L IH N T AH N/`→ `/k` *e* `l in d un/`

`[rule 2] Duplicate nasals whenever they are surrounded by vowels.`
`Diana`→`/D AY AE` *N* `AH/`→`/D AY AE` *N N* `AH/`→`/d ai` *an n* `a/`

could be easily undermined by the following exceptions, respectively:

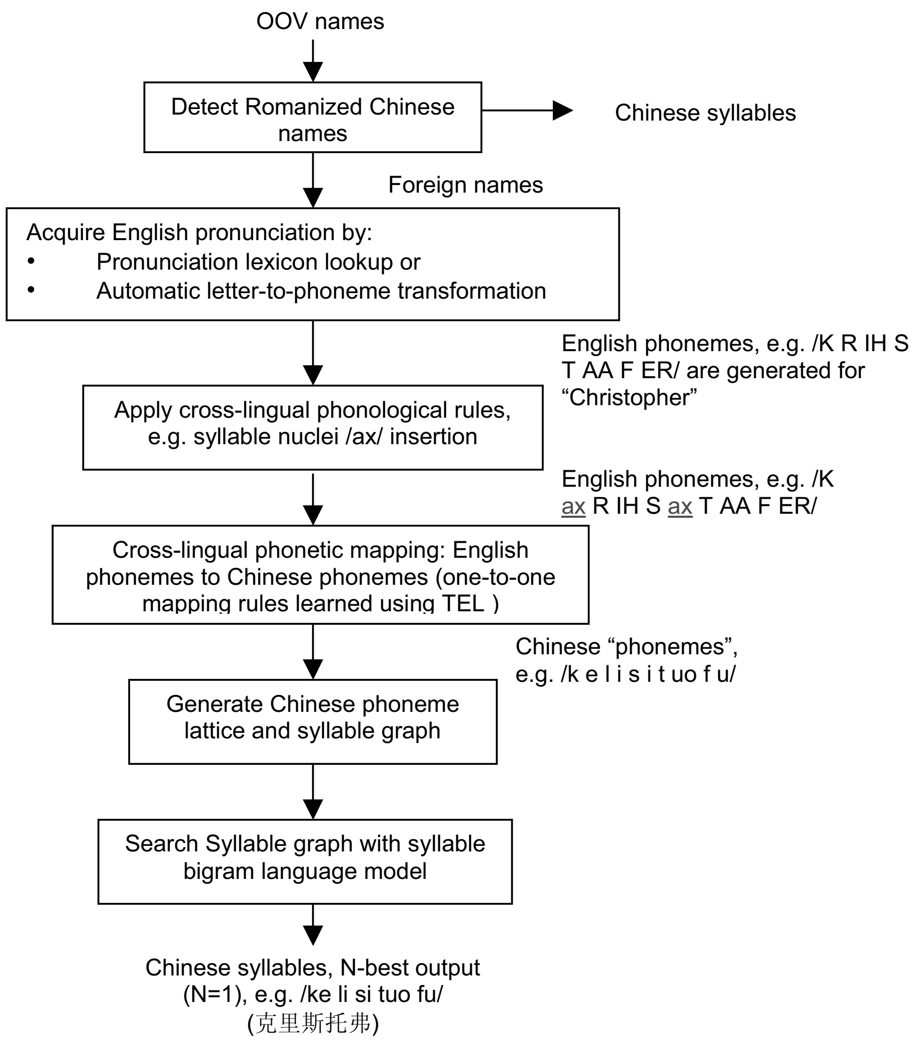
OOV names
Detect Romanized Chinese names
Chinese syllables
Foreign names
Acquire English pronunciation by:
• Pronunciation lexicon lookup or
• Automatic letter-to-phoneme transformation
English phonemes, e.g. /K R IH S T AA F ER/ are generated for "Christopher"
Apply cross-lingual phonological rules, e.g. syllable nuclei /ax/ insertion
English phonemes, e.g. /K ax R IH S ax T AA F ER/
Cross-lingual phonetic mapping: English phonemes to Chinese phonemes (one-to-one mapping rules learned using TEL)
Chinese "phonemes", e.g. /k e l i s i t uo f u/
Generate Chinese phoneme lattice and syllable graph
Search Syllable graph with syllable bigram language model
Chinese syllables, N-best output (N=1), e.g. /ke li si tuo fu/ (克里斯托弗)

Figure 2.1: The transliteration process in [23]

```
[exception 1] Clint →/K L IH N T/→/k e l in t e/

[exception 2] Canada→/K AE N AH D AH/→/j ia n a d a/
```

Unlike the consonant cluster /K L/ obeying rule 1, where an /ax/ should be inserted between them, such an insertion is not required in the similar cluster of /N T/ in the example of exception 1. Likewise, the nasal /N/ surrounded by vowels also need not to be duplicated in the example of exception 2 as opposed to the stipulation of rule 1.

In summary, this method, which was not presented as a model in [23], could be generalized using the following equation:

$$\operatorname*{argmax}_T \Pr(T|S) = \operatorname*{argmax}_T \{\Pr(T|S) \times \Pr(T)\} \tag{2.3}$$

This equation is beyond Beyes' theorem. Eq.(2.3) is basically a *direct* transliteration model which will be discussed in detail in later chapters.

2.2.4 Source-channel-based Approach

Based on the source-channel framework, [33] described a fully data-driven approach for English-to-Chinese transliteration using the state-of-the-art statistical machine translation (IBM SMT) model [7]. We will compare our work closely with [7] as their system adopts widely recognized model and their research objectives are similar to ours.

The IBM SMT model is based on the well-known source-channel framework, which was initially tested for French-to-English machine translation. The model was intensively studied by quite a number of researches in machine translation [1, 12, 27, 25]. When applied to English-to-Chinese transliteration, the fundamental equation is as follows:

$$\begin{aligned}\hat{C} &= \operatorname*{argmax}_C \Pr(C|E) = \operatorname*{argmax}_C \{\Pr(E|C) \times \Pr(C)\} \\ &= \operatorname*{argmax}_{c_1^{|C|}} \left\{p_\theta(e_1^{|E|}|c_1^{|C|}) \times p_\gamma(c_1^{|C|})\right\} \end{aligned} \tag{2.4}$$

where $E = e_1^{|E|}$ denotes a $|E|$-phoneme English word as the observation on channel output, and $C = c_1^{|C|}$ represents E's $|C|$-phoneme Chinese translation by pinyin as the source of channel input. The channel decoder reverses the direction, i.e. to find the most probable input pinyin sequence given an observation E. The posterior probability $\Pr(C|E)$ is indirectly maximized by optimal combination of the transliteration model $\Pr(E|C)$ and the language model $\Pr(C)$. We use $p(.)$ to denote model-based probability distributions

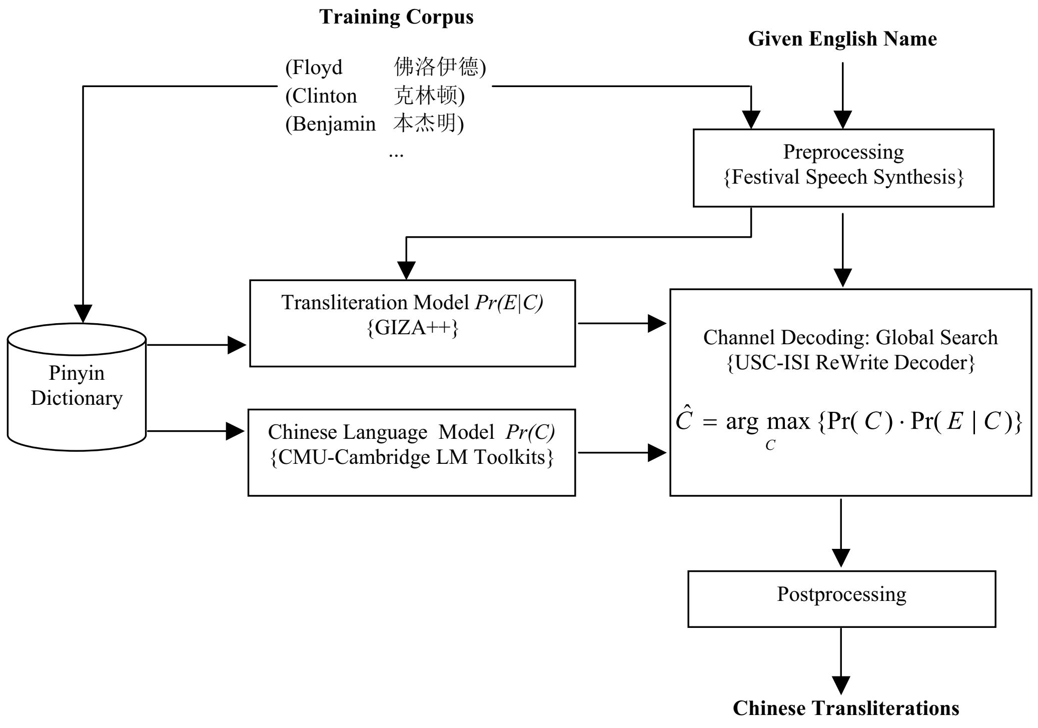
Training Corpus
(Floyd 佛洛伊德)
(Clinton 克林顿)
(Benjamin 本杰明)
...
Given English Name
Preprocessing {Festival Speech Synthesis}
Pinyin Dictionary
Transliteration Model Pr(E|C) {GIZA++}
Chinese Language Model Pr(C) {CMU-Cambridge LM Toolkits}
Channel Decoding: Global Search {USC-ISI ReWrite Decoder}
$\hat{C} = \arg\max_C \{\Pr(C) \cdot \Pr(E \mid C)\}$
Postprocessing
Chinese Transliterations

Figure 2.2: English-to-Chinese transliteration system [33] based on IBM SMT model

with particular assumptions in contrast to the general probability distributions Pr(.). θ and γ are the parameters associated with the two models respectively.

The transliteration process is illustrated in Figure 2.2. The transliteration model $\Pr(E|C)$ was trained from name pairs represented in International Phonetic Alphabet (IPA) symbols at the English side and pinyin notations at the Chinese side. Standard bootstrapping of the IBM translation models using GIZA++[1] [1] was applied in the training process. Specifically, 5 *EM* iterations of Model-1 followed by 5 of Model-2, 10 of Model-HMM and 10 of Model-4 were used. The Language model $\Pr(C)$ was trained on pinyin vocabulary using *trigram* of pinyin symbols with *Good-Turing smoothing* and *Katz back-off* provided by CMU-Cambridge Language modeling toolkits[2] [9]. Searching was done by using USC-ISI ReWrite Decoder[3] [12]. These are standard packages for IBM SMT model training and testing, which will be

[1]http://www.isi.edu/~och/GIZA++.html
[2]http://mi.eng.cam.ac.uk/~prc14/tookit.html
[3]http://www.isi.edu/license-sw/rewrite-decoder/

System	Training Size	Test Size	Pinyin Errors
[23]	2233	1541	52.5%
Small MT [33]	2233	1541	50.8%
Big MT [33]	3625	250	49.1%

Table 2.1: Comparisons presented by [33] on error rates between transliteration systems, namely [23] and [33]

briefly introduced in the next chapter. Note that Festival speech synthesis system[4] was employed to convert English names into their corresponding phonetic representations.

Experimental Study

The method proposed by [23] was compared with this system by [33] using the same data set. The performance was measured by pinyin error rates with edit distance. The result is shown in Table 2.1. There are at least three noticeable observations from the experiment:

1. Small MT and [23] shared the same data set for training and testing. Source-channel approach showed slightly better results. However, since the two systems employed different letter-to-phoneme generators, it was unclear whether they distinguished due to grapheme-to-phoneme generation, or the phoneme transliteration process or both; another possible reason might be different language models used, i.e. syllable bigram by [23] and symbol trigram by [33], but it was not conclusive since the two language models differed not only from length of grams n, but also from level of grams considered, i.e. syllable or sub-syllable (symbol).

2. Better performance was observed in Big MT. Big MT is based on the same implementation as Small MT, except a different and larger training set and a smaller testing set. Thus it was evident that a larger training set could probably yield better results. But this shallow comparison could not provide useful guidelines for identifying and improving the intrinsic deficiencies in the underlying model.

3. Edit distance based measurement may be not effective enough to differentiate the performance of the two transliteration systems since the criteria is relatively loose. Even though the edit distance error of the

[4]http://fife.speech.cs.cmu.edu/festival/

two systems is close, true performance on accuracy may be even larger. We will testify this supposition in later chapters.

Other Remarks

Back to the source-channel model, it was initially proposed for backward transliteration from Japanese to English origins [19], in which the reversed prior probability $\Pr(J|E)$ can naturally coincide with the original direction in producing transliterations by human translators, i.e. the model still concentrates on how to produce J given E origins. When the model is applied to forward transliteration, however, it should be noted that the conditional probability $\Pr(E|C)$ in Eq.(2.4) is an opposite estimation of actual transliteration. Conditioning on C, one has to consider how probable to produce each given pinyin symbol from certain English phoneme(s). It is intuitively unnatural and more seriously, error-prone for forward transliteration due to the difficulties of identifying mapping relationships conditioning on individual target symbols (see Section 3.2).

2.3 Chapter Summary

We have summarized the previous techniques for machine transliteration. We paid much attention to source-channel based and other statistical approaches, particularly on the contributions of [23] and [33] for English-to-Chinese transliteration. Statistical approaches have obvious advantages: They are more extendable as they are data-driven, and can remove ad hoc procedures in rule-based methods; they can readily tolerate existing transliteration discrepancies; they are capable of both *producing* transliterations and *recognizing* transliterations compared to similarity-based approaches. We highlighted the system applying the IBM SMT model. We will proceed to use this well-known model as our baseline.

□ **End of chapter.**

Chapter 3

Transliteration Baseline

Before we proceed to propose a new transliteration model. We will analyze the widely used IBM statistical machine translation (IBM SMT) system and identify its limitations in English-to-Chinese name transliteration. In this chapter, we replicate the system described by [33] using IBM SMT. This implementation provides the baseline for our research.

3.1 Transliteration Using IBM SMT

3.1.1 Introduction

IBM SMT model is a well-established language-independent probabilistic framework for translating a sentence from a source to a target language according to the statistical translation relations acquired from bilingual corpora. SMT views any target language word as a potential translation of any word in the source language [1]. The probability distribution in Eq.(2.2) is over all pairs of words. Given S, the model can output T which maximizes $\Pr(T|S)$. The priori $Pr(S|T)$ (i.e. the translation model) works for ensuring that T is normally interpreted as S but not others, while $P(T)$ (i.e. the language model) ensures the output T natural and grammatical. Training algorithms are required to fix two models' parameters, and a decoding algorithm is needed for searching the most probable T. The algorithms are implemented in GIZA++ for translation model training, CMU-Cambridge toolkits for language model training and USC-ISI Rewrite Decoder for searching.

Simply and uncritically using the SMT toolkits, [33] didn't relate the model with enough information concerning its application to English-to-Chinese transliteration. It is unclear about its operations over phonemes and its appropriateness for this yet another different task. We therefore attempt to identify the limitations of the model for this application and provide useful guidelines for improvements. In this study, S and T correspond to English and Chinese respectively, and sentences for translation are reduced to phoneme sequences and words in sentences are reduced to phonetic units (see Eq.(2.4)).

3.1.2 GIZA++ for Transliteration Modeling

GIZA++ [25, 27] is an extension of the GIZA program [1] (part of the SMT toolkit EGYPT[1]). We applied the toolkit to transliteration model training. GIZA++ reversely considers how transliteration can be done from target pinyin symbol sequences back to source English phoneme sequences. Here source and target refer to Chinese and English respectively. The toolkit implements the SMT alignment models of Model-1 through Model-4 described in [7] and Model-HMM in [31], in which the training is typically carried out as a bootstrapping process starting from a simple model toward more complex models to iteratively improve alignments between parallel phoneme sequences until the *EM* algorithm converges to the *Viterbi* alignment of all pairs [1, 18, 33]. These models differ from alignment details and parameters in terms of the correspondence defined between phonetic units in the pronunciation sequences of given name pairs.

Alignment Scheme

We first exemplify Model-3[2] to illustrate the typical pairwise alignments and types of dependencies provided in IBM SMT model. A good alignment over a pair of phoneme sequences that obtained from GIZA++ training is shown in Table 3.1[3]. Conditioning on the pinyin side, each English phoneme is aligned to only one pinyin symbol. The numbers are the positions of phonetic symbols in the respective sequences. If a pinyin symbol has ϕ English phonemes aligning to it, it has a *fertility* of ϕ. If it remains unaligned to any English phoneme, it is known as *zero-fertility*. Likewise, if an English phoneme is left unaligned, it is called *NULL-generated*. For example, /i/ is zero-fertility, which is aligned to the mute ε, /an/ has fertility 2, other pinyin symbols have fertility 1 and /D/ is NULL-generated, which is aligned to the NULL symbol. Note that position 0 is reserved for all NULL symbols.

[1]http://www.clsp.jhu.edu/ws99/projects/mt/toolkit/

[2]Although Model-3 currently is not supported by GIZA++ any longer, the understanding of this typical model helps know all the others. We hence exemplify it here.

[3]Lowercase letters denote pinyin symbols. Capitalized letters are English phonemes represented by computer-readable IPA notations-APRABET.

Stanford							
1		2	3	4	5	6	7
S	ε	T	AE	N	F	ER	*D*
s	*i*	t	an		f	u	*null*
1	2	3	4		5	6	0
斯坦福							

Table 3.1: Example of Phoneme Alignment in Model-3

To produce the corresponding English phoneme sequence of a pinyin symbol sequence without knowing the alignment in advance, a stepwise decision-making process is used. The process assumes a set of model parameters:

- C: Chinese pinyin symbol sequence.
- E: English phoneme sequence.
- c_i: The ith pinyin symbol.
- e_j: The jth English phoneme.
- l: The number of symbols in the pinyin sequence.
- m: The number of phonemes in English phoneme sequence.
- A: Alignment between C and E, a vector of integers $A = \{a_1, a_2, \ldots, a_m\}$ where $0 \leq a_j \leq l$.
  Example: $A = \{1, 3, 4, 4, 5, 6, 0\}$ given the alignment in Table 3.1.
- a_j: The pinyin position connected to by the jth English phoneme in alignment A.
  Example: $a_1 = 1, a_2 = 3, a_3 = a_4 = 4, a_5 = 5, a_6 = 6, a_7 = 0$
- c_{a_j}: The actual pinyin symbol connected to by the jth English phoneme in alignment A.
  Example: $c_{a_3} = c_{a_4} = /an/$
- ϕ_i: Fertility of pinyin symbol c_i where $1 \leq i \leq l$, given the alignment A.
- ϕ_0: Fertility of NULL symbol, also be the number of NULL inserted.
- $n(\phi_i|c_i)$: n table—fertility probability of pinyin symbols.

- p_1, p_0: p table—probability of inserting or NOT inserting NULL after a pinyin symbol. $p_0 = 1 - p_1$.
- $t(e_j|c_{a_j})$: t table — probability of translating the pinyin symbol to the English phoneme under the alignment A.
- $d(j|a_j, l, m)$: d table — Distortion probability of English phonemes not generated by NULL. j is the target phoneme position, a_j is the position in the pinyin sequence of the symbol that generated the jth English phoneme now being placed, given the alignment A.

The decision-making process proceeds as follows:

1. Every pinyin symbol c_i in C is assigned a fertility ϕ_i according to probability table $n(\phi_i|c_i)$. $\sum_{i=1}^{|C|} \phi_i = \bar{m}$.
   Example: $/s/ \preceq 1, /i/ \preceq 0, /t/ \preceq 1, /an/ \preceq 2, /f/ \preceq 1, /u/ \preceq 1$("$\preceq$" denotes fertility) and $\bar{m} = 6$, or other possibilities accordingly.
2. Make a new C sequence by deleting pinyin symbols with fertility zero, copying symbols with fertility one, and duplicating symbols with fertility two, etc.
   Example: /s *i* t an f u/→/s t *an an* f u/ or other possibilities accordingly.
3. After producing each pinyin symbols in the new sequence, make a decision to insert ϕ_0 number of NULL symbols with probability p_1 or not (insert) with probability p_0. The total number of C symbols then becomes $m = \bar{m} + \phi_0$. The NULL symbols will finally be used to produce NULL-generated English phonemes.
   Example: /s t an an f u/→/s t an an f u *null*/. Note that the insertion is attempted at every possible position accordingly.
4. Perform replacement of each symbol in C with an English phoneme according to the probability table $t(e_j|c_{a_j})$, including NULL symbols.
   Example: /s t an an f u *null*/→/S T AE N F ER *D*/ or other possibilities accordingly.
5. Assign English phoneme positions to those phonemes not generated by NULL according to the probability table $d(j|a_j, l, m)$. Note that this step is for sentence translation and may be ineffective since the order of phoneme pairs of the two languages are strictly sequential without distortion.

6. Assign English phoneme positions for the NULL-generated phonemes. They should fall into ϕ_0 empty positions with equal probability $1/\phi_0!$.

7. Output the generated English phoneme sequence.

These steps are modeled as a generative stochastic process. It starts from a given C and results in different choices of E as well as different alignments A of E with C. The probability $p(E|C)$ is the summation over all possible alignments A:

$$p(E|C) = \sum_A p(E,A|C) \tag{3.1}$$

$p(E,A|C)$ is symbolized as the following probability by Model-3 [1, 7, 18]:

$$\begin{aligned} p(E,A|C) = & \prod_{i=1}^{l} n(\phi_i|c_i) \prod_{i=0}^{l} \phi_i! \times \\ & \binom{m-\phi_0}{\phi_0} p_0^{m-2\phi_0} p_1^{\phi_0} \times \\ & \sum_{j=1}^{m} t(e_j|c_{a_j}) \times \\ & \frac{1}{\phi_0!} \times \\ & \prod_{j:a_j \neq 0}^{m} d(j|a_j,l,m) \end{aligned} \tag{3.2}$$

where the factors separated by $\times$ denote fertility, NULL-insertion, translation, distortion of NULL-generated phonemes and distortion of non-NULL generated phonemes in terms of the generative decisions above. In this regard, Model-3 is a zero-order dependency and many-to-one (from source to target phonetic units) stochastic alignment model allowing for distortion of target symbol positions.

Other models are somewhat different from Model-3, which are briefly introduced as follows:

- Model-1 only uses a t table, and is a zero-order dependency and one-to-one alignment model, assuming a uniform alignment probability [7]:

$$p(E,A|C) = \frac{1}{(l+1)^m} \prod_{j=1}^{m} t(e_j|c_{a_j})$$

- Model-2 is also zero-order HMM and one-to-one alignment model, which uses t table and alignment probability $d(a_j|j,l,m)$ instead of distortion probability $d(j|a_j,l,m)$ [1, 7]:

$$p(E,A|C) = \prod_{j=1}^{m} t(e_j|c_{a_j})a(a_j|j,l,m)$$

- Model-HMM is a first-order HMM and one-to-one alignment model, which uses t table and assumes alignment position a_j depends only one its previous alignment position a_{j-1} [31]:

$$p(E,A|C) = \prod_{j=m}^{m} t(e_j|c_{a_j})p(a_j|a_{j-1},l,m)$$

- Model-4 is extended from Model-3, which does not use distortion table d, but instead differentiates permuting English phonemes that are heads, non-head and NULL-generated. The permutation encourages the adjacent pinyin symbols to translate into adjacent English phonemes. This is more appropriate than Model-3 which allows for distortions required in translation but unnecessary in transliteration. However, it is also a zero-order dependency and many-to-one alignment model. See [7, 12] for details.

Remarks

The model uses many stochastic parameters, catering for the randomized presentation and positioning of unaligned words in machine translation. However, these parameters and operations may be ineffective when the model is applied to transliteration. We noticed that the following operations tend to be inaccurate: (a). the zero-fertility pinyin symbols are deleted in step 2, which should be somehow (actually randomly) reproduced during decoding stage; (b). ϕ_0 NULL symbols are inserted with the probability p_0 in step 3. The positions for insertion are determined in terms of parameters ϕ_0, p_0 and p_1 by probing every possible position in the source sequence; (c). each NULL-generated English phoneme in step 4 is assigned one of the ϕ_0 positions with probability $1/\phi_0!$ in step 6. Then we would naturally raise the question: if the appearances and positions of unaligned pinyin symbols and English phonemes were not stochastic, where should they practically go? Or what could be a more precise decision-making process?

3.1.3 CMU-Cambridge Toolkits for Language Modeling

The CMU-Cambridge language modeling toolkit was released for facilitating the construction and testing of n-gram language models. It is currently used by many institutions worldwide. Its usage details can be found in [9].

We use the toolkit for training the target language model based on trigrams of pinyin sub-syllables (initials and finals, see Section 3.3.1). The maximum likelihood estimate is biased high for observed grams and biased low for unobserved ones. To correct this bias, smoothing techniques are used to redistribute some probability mass from the observed grams to the unseen ones. Good-Turing discounting and Katz back-off [9, 33] are applied for redistribution of probability mass toward unobserved trigrams in the training data.

3.1.4 ReWrite Decoder for Decoding

Rewrite Decoder includes probabilistic decoding algorithms used to yield the most likely machine translations according to the previously trained parameters of the translation model and language model. Given a new English phoneme sequence, the decoder searches for the pinyin symbol sequence that maximizes $p(C|E)$ in terms of Eq.(2.4) where $p(E|C)$ is the summation of $p(E, A|C)$ over all alignments A shown as Eq.(3.1). Because the sum involves significant computations, it is typically avoided by instead search for an $< C, A >$ pair that maximizes $p(C, A|E) \simeq p(E, A|C) \cdot p(C)$ [12].

The solution was built incrementally by applying operations to each subset of input phonemes and iteratively testing partial hypotheses generated according to the parameters previously learned until optimality/sub-optimality is reached. There are four possible operations [12]:

- **Add** adds a new pinyin symbol and aligns a single English phoneme to it.
  Example: /S/→/s/.
- **AddZfert** adds two new pinyin symbols. The first has fertility zero, while the second is aligned to a single English phoneme.
  Example: /ε/→/i/ + /T/→/t/.
- **Extend** aligns an additional English phoneme to the most recent pinyin symbol so as to increase its fertility.
  Example: /AE + N/→/an/.

- **AddNull** aligns an English phoneme to the pinyin NULL symbol.
  Example: /D/→/null/.

AddZfert considers inserting a zero fertility pinyin symbol before each transliteration of each newly unaligned English phoneme [12]. **AddNull** considers each input phoneme as possible NULL-generated phoneme aligned to the NULL symbol. This is because the model believes that it is impossible to know the positions in advance where zero-fertility pinyin symbols and NULL-generated English phonemes should appear. The decision is left for the optimization process, in which the partial hypotheses incrementally maximize the posterior probability using a hill-climbing strategy participated by the transliteration model and the language model.

3.2 Limitations of IBM SMT

IBM SMT model was designed for machine translation. It has several limitations for English-to-Chinese transliteration noticeably:

Problem 1: The model has a tight constraint on mapping relationship between the source and target words. It allows only one target language word to be associated with a contiguous group of source language words, but not vice versa. As such in English-to-Chinese transliteration, one English phoneme can never be converted to a group of Chinese pinyin symbols. The limitation results from the difficulty due to conditioning on C in the *inverted* conditional probability $\Pr(E|C)$ as the transliteration model is unable to detect possible contiguous combinations of target phonemes prior to training.

The example in [33] exposes this obvious limitation (see Figure 3.1). Because of the restriction, /u/ and the second /i/ in the third line have to be considered as *zero-fertility* "words". Typical zero-fertility pinyin symbols collected by GIZA++ training include $\{i, e, u, o, ou, ie, \cdots\}$, which are finals that usually form syllables with their previous syllable initials. In fact, such syllables are initial-final *clusters* which are aligned to single English phonemes, such as /F/→/fu/ and /S/→/si/ in the example. Zero-fertility symbols are "deleted" by source-channel during training and stochastically "reproduced" during decoding. Reproduction is done by the **AddZfert** operation. It inserts a possible zero-fertility symbol before each target symbol of each remaining unaligned source phoneme and it needs to consider all the possible zero-fertility symbols. To reduce the cost of **AddZfert**, two approximations are adopted [12]:

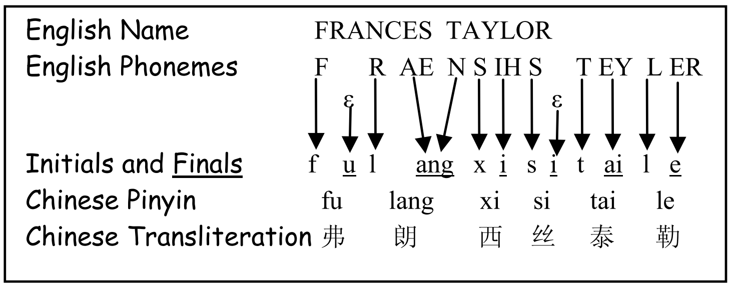

Figure 3.1: The English-to-Chinese transliteration example using IBM SMT model in [33]

1. Only consider certain pinyin symbols as candidates for zero-fertility, namely symbols which both occur frequently and have high probability of being assigned fertility zero.
2. Only insert a zero-fertility symbol if it will increase the probability of a hypothesis.

We observe that zero-fertility symbols are unavoidable in source-channel since the model does not allow for one-to-many mappings from source to target symbols, which results from the reversed priori conditional probability. Because the prediction for zero-fertility position is entirely stochastic, the model would be less capable of predicting zero-fertility finals that are very likely sticking to their preceding initials.

Problem 2: Due to smoothing, the language model may not assign zero probability to an illegal pinyin sequence that is unobserved in the training data, e.g. one containing two consecutive initials. Such sequences are required to be corrected by inserting suitable finals between them until a legitimate pinyin sequence is obtained [33]. Although this could fix illegitimate pinyin, it would produce wrong transliterations as the insertion of a final has no probabilistic basis.

We observe that consecutive initials can come from the model stochastically predicting positions for zero-fertility symbols. For instances, for /K L IH N T IN N/ (Clinton) which should be transliterated into /ke lin dun/ (克林顿), if the model is unable to correctly predict that there should be a zero-fertility /e/ inserted between /k/ and /l/ in the transliteration, the consecutive initials /kl/ would result. This would leave the correction be done by groundless insertion trials since the language model also accepts it with some probability mass.

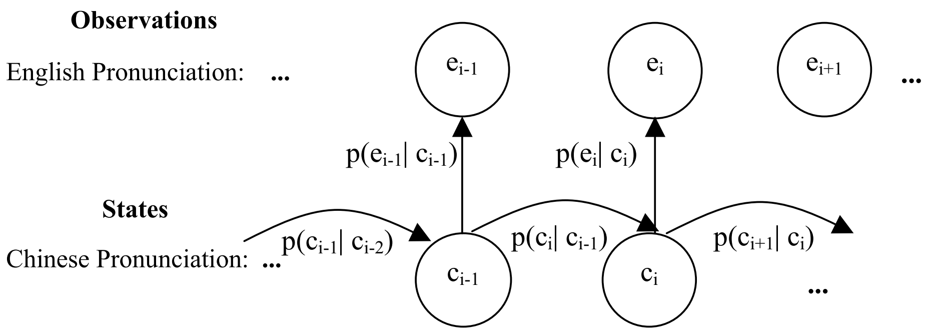

Figure 3.2: Transliteration model is implemented under two assumptions of Markov model

Problem 3: Transliteration model $\Pr(E|C)$ is approximated by Markov chains [29]. Markov model is implemented primarily under two assumptions: Markov assumption (first-order or second-order) on state transition and conditional independence assumption on observation. This is illustrated in Figure 3.2. Markov assumption hypothesizes that transition probability to a state (i.e. Chinese pinyin symbol), depends only on its immediate previous one or two states. Conditional independence assumption assumes that an observation unit (i.e. English phoneme), depends only on the state (i.e. Chinese pinyin symbol) that generates it, and not on its neighboring observation units.

With these assumptions, it is hard to extend the model with additional dependencies [26], such as features of neighboring phonemes on both sides. Albeit the trigram language model $Pr(C)$ is combined, it can only make use of a short history in the target language context. One may consider to use longer distance dependencies in $\Pr(E|C)$, and it is possible to break down the longer history in the conditional probability into smaller fragmented probability terms in order to alleviate its vulnerability to data sparseness. However, one has to assume certain dependencies or independencies for this case in terms of heuristics, e.g. longer fragments could be substituted by shorter ones as redundant terms exist in longer fragments, or some terms is farther from the current prediction position than others.

Problem 4: Since the training of the language model is independent of the transliteration model, their combination sometimes may yield unpredictable results. Empirically, Eq.(2.4) cannot be optimal unless true probability distribution of the two individual portions are used [26]. Yet the used models and trained methods in machine translation only

GIZA++ *EM* Itera.		CMU-Camb. LM		ReWrite Decoder	
Model-1:	5	n-gram:	3	search:	fast greedy
Model-2:	5	discounting:	Good-Turing		
Model-HMM:	10	back-off:	Katz		
Modle-4:	10	forced back-off:	<s>,</s>		

Table 3.2: Experimental settings to the IBM SMT toolkits

Consonants (24)	P	T	K	B	D	G	M	N
	NG	F	V	TH	DH	S	Z	SH
	ZH	CH	JH	L	W	R	Y	HH
Vowels (16)	IY	IH	EY	EH	AE	ER	AH	AX
	AY	AW	AA	OW	OY	AO	UW	UH

Table 3.3: 40 symbols in the ARPABET pronunciation inventory

provided poor approximations of true distributions [26]. This implies that it is difficult for the transliteration model to achieve optimality. Therefore, a different combination of language model and transliteration model might be more effective or ineffective.

3.3 Experiments Using IBM SMT

We carried out experiments to evaluate IBM SMT toolkits for transliteration performance. Experimental settings on the model were the same as [33], which is listed in Table 3.2. But we excluded the letter-to-phoneme and Pinyin-to-Hanzi modules (see [33]). The omission helps identify errors produced by the phoneme-based transliteration.

3.3.1 Data Preparation

The Phonetic Representation

Pronunciation sequences of English names are represented by computer-readable IPA equivalents, ARPABET symbols, for American English. There are 40 APRPABET symbols in total, in which 24 are consonants and 16 are vowels as shown in Table 3.3. The vowel phoneme /AX/ is known as *nuclei* or *schwa*, and generally does not appear in real corpus. Thus the number of symbols actually being used would be 39. The correspondence between APRABET symbols and IPA symbols is listed in Appendix A.

Initials (23)	b	p	m	f	d	t	n	l	g	k	h	j
	q	x	zh	ch	sh	r	z	c	s	y	w	
Finals (35)	a	o	e	ai	ei	ao	ou	er	an	en	ang	eng
	i	ia	ie	iao	iu	ian	in	iang	ing	iong	u	ua
	uo	uai	ui	uan	un	uang	ong	ü	üe	üan	ün	

Table 3.4: 58 GIF symbols in pinyin Romanization system

Chinese transliteration is phonetically represented by pinyin symbols using the common Romanization system for Mandarin Chinese. Pinyin are composed of 23 initials and 35 finals, which are also referred to as generalized initial and finals (GIF) in [33]. Table 3.4 shows these symbols. Note that the symbol "ü" is non-ASCII, it is substituted by the symbol "v" internally.

Corpus

We obtained the beta release v.1.0 of LDC's Chinese-English bi-directional named entity list[4] compiled from Xinhua's database. The entire corpus consists of 9 pairs of lists, from which we chose the English-to-Chinese proper name list of people as raw data. The list contains 572,213 foreign people's names and their Chinese transliterations. Note that although the list is in English, it contains names originated from different languages, e.g. Russian, German, Spanish, Arabic, Japanese, Korean, etc. One assumption is that the Chinese translations were produced based on their English pronunciations directly. The exceptions are Japanese and Korean names, which are generally translated in terms of meaning as opposed to pronunciation. We consider these names as noisy data.

We used CMU's pronunciation dictionary[5] and LDC's Chinese character table with pinyin to convert name pairs in the list into a parallel corpus of English phonemes and pinyin symbols. We extracted all the translation name pairs from the selected named entity list, which also appeared in CMU pronunciation dictionary with deterministic phonetic representations. We then obtained their English pronunciations and the pronunciations of their Chinese equivalents by looking up the pronunciation dictionary and the character-pinyin conversion table. We ended up with 46,305 pairs, which were used as our experimental data pool. In our experiments, 90% instances (41,674 instances) were randomly selected for training, in which a portion of 4,631 instances were used for close test, and the remaining 10% (4,631 instances) for open test. The transliteration model and language model were trained on

[4]Catalog Number by Linguistic Data Consortium: LDC2003E01

[5]ftp://ftp.cs.cmu.edu/afs/cs.cmu.edu/data/anonftp/project/fgdata/dict/

C.A.		W.A.	
Close	Open	Close	Open
66.35%	65.15%	20.73%	18.27%

Table 3.5: Experimental results by our implementation of [33]'s transliteration system

the same 41,674 instances, in which the Chinese part of the parallel corpus is used to train language model and the entire parallel corpus for transliteration model. This data set is referred to as *Base-0*, which was used in our comparative experiments.

3.3.2 Performance Measurement

There is no existing criterion for measuring machine transliteration accuracy. Some tests require human judgment [1, 19]. The performance was evaluated with two levels of accuracy, i.e. character-level accuracy (*C.A.*) and word level accuracy (*W.A.*) in [17].

$$C.A. = \frac{L - (i + d + s)}{L} \tag{3.3}$$

$$W.A. = \frac{\#\ of\ correct\ names\ generated}{\#\ of\ tested\ names} \tag{3.4}$$

In Eq.(3.3), L is the length of the standard transliteration of a given foreign name, and i, d, and s are the number of *insertion*, *deletion* and *substitution* respectively, i.e. edit distance between machine-generated transliteration and the standard. If $L < (i + d + s)$, we set $C.A. = 0$. Eq.(3.4) is the percentage of the number of transliterations identical to the standards in all the tested names. Here we prefer to use "identical" rather than "correct" because the standard transliterations are de facto rather than absolute.

3.3.3 Experimental Results

The experimental results of close and open tests on data set *Base-0* measured by *C.A.* and *W.A.* is shown in Table 3.5.

We note that the *C.A.* of our implementation is reasonably higher in our tests than that reported in [33] where they achieved the pinyin errors in edit distance by 42.5%∼50.8% (see Table 2.1), corresponding to 49.2%∼59.5% if using our *C.A.* measure. Although the discrepancies are mainly due to the different data sets, one of the possible reasons leading to their lower accuracy is the use of letter-to-phoneme generation system in [33] whereas we used pronunciation dictionary. We will use the results as the baseline for comparing with our models.

3.4 Chapter Summary

This chapter first analyzed the application of IBM SMT system to English-to-Chinese transliteration in order to reveal the limitations the source-channel model. Before proposing our direct models to overcome such limitations, we have replicated the phoneme transliteration system described by [33] in order to compare the performance between our contribution and the source-channel based approach. This implementation also serves as the baseline for comparative experiments in later chapters.

□ **End of chapter.**

Chapter 4

Direct Transliteration Modeling

The source-channel based transliteration model fails to correctly handle mapping probabilities of zero-fertility symbols in the target names (see Section 3.2). In Figure 3.1, it would be more natural and easier to handle if /f-u/ and /s-i/ were treated as initial-final *cluster* when they were converted from the single English phonemes /F/ and /S/, respectively.

Different from the source-channel model, we propose to estimate the posterior probability directly. We adopt a different angle of observation to avoid the use of the reversed conditional probability, i.e. we propose to condition on E rather than on C. As such, Figure 4.1 shows the application of the alignment scheme of our approach to the previous example in Figure 3.1. Notice that combination of pinyin symbols are regarded as initial-final clusters, e.g. /F/ to /fu/ and /S/ to /s/.

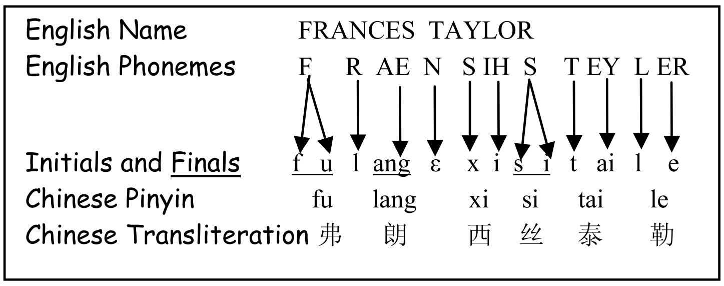
English Name FRANCES TAYLOR
English Phonemes F R AE N S IH S T EY L ER
Initials and Finals f u l ang ε x i s i t ai l e
Chinese Pinyin fu lang xi si tai le
Chinese Transliteration 弗 朗 西 丝 泰 勒

Figure 4.1: The phoneme alignment scheme in direct transliteration modeling

4.1 Soundness of the Direct Model—Direct-1

We substitute $\Pr(E|C)$ by $\Pr(C|E)$ in Eq.(2.4) resulting in the following transliteration method:

$$\begin{aligned}\hat{C} &= \underset{C}{\operatorname{argmax}} \Pr(C|E) = \underset{C}{\operatorname{argmax}} \{\Pr(C|E) \times \Pr(C)\} \\ &= \underset{c_1 c_2 \ldots c_{|C|}}{\operatorname{argmax}} \left\{ p_\theta(c_1^{|C|}|e_1^{|E|}) \times p_\gamma(c_1^{|C|}) \right\} \end{aligned} \tag{4.1}$$

$\Pr(C|E)$ aims to produce the most likely transcriptions for a given E, but ill-formed pinyin sequences may result. The language model $\Pr(C)$ is introduced to make correction, e.g. eliminating illegal pinyin strings and yielding better ranking of the resulting syllables. Although Eq.(4.1) is beyond the Bayes' theorem, it is mathematically sound under the more general *Maximum Entropy* (MaxEnt) framework [4, 26].

MaxEnt is a well-founded framework for directly modeling the posterior probability, where a set of M feature functions $f_m(E, C)$ and their corresponding model parameters λ_m ($m = 1, \ldots, M$) are introduced. According to [26], direct transliteration probability can be approximated by:

$$Pr(C|E) \simeq p_{\lambda_1^M}(C|E) = \frac{\exp\left\{\sum_{m=1}^{M} \lambda_m \cdot f_m(E, C)\right\}}{Z_{\lambda_1^M}(E)} \tag{4.2}$$

where the denominator

$$Z_{\lambda_1^M}(E) = \sum_{C'} \exp\left\{\sum_{m=1}^{M} \lambda_m \cdot f_m(E, C')\right\}$$

is a normalizing constant determined by the requirement that $\sum_{C'} p_{\lambda_1^M}(C'|E) = 1$ for all E. C' denotes all possible Chinese transliterations for the given E.

The computation for the normalizing constant is very time-consuming, but it is not required for maximization (search) process [26]. We could then obtain the target sequence $\hat{C}$ that maximizes the posterior probability by omitting the denominator:

$$\hat{C} = \underset{C}{\operatorname{argmax}}\, Pr(C|E) = \underset{C}{\operatorname{argmax}} \left\{ \exp\left[\sum_{m=1}^{M} \lambda_m \cdot f_m(E, C)\right]\right\} \tag{4.3}$$

We can select two feature functions and their parameters:

$$\begin{aligned} f_1(E, C) &= \log p_\theta(C_1^{|C|}|E_1^{|E|}) \\ f_2(E, C) &= \log p_\gamma(C_1^{|C|}) \\ \lambda_1 &= \lambda_2 = 1 \end{aligned}$$

Thus, Eq.(4.1) obtains in combination of direct transliteration model $p_\theta(c_1^{|C|}|e_1^{|E|})$ and target language model $p_\gamma(c_1^{|C|})$ with respect to model parameters θ and γ. The optimal parameters are estimated individually from the parallel training corpus. This model is referred to as **Direct-1**.

4.2 Alignment of Phoneme Chunks

There are 4 possible general conditions, in which an English phoneme can map to items in the pinyin's vocabulary in the direct model:

1. An English phoneme maps to an initial or a final, which is the most usual case;
2. An English phoneme maps to an initial-final cluster, e.g. /F/–/fu/ and /S/–/si/ in the previous example;
3. An English phoneme maps to a mute ε, e.g. /S T AE **N** F ER **D**/ (Stanford) to /si tan fu/ (斯坦福), where /N/ and /D/ are omitted in translation;
4. Insert additional pinyin syllables, e.g. /F L OY D/ (Floyd) to /fu luo **yi** de/ (佛洛伊德), where /yi/ is inserted to cater for the sound /OY/ that has already been mapped to /uo/.

We introduce alignment indicators between a pair of sound sequences, E and C. Within 39 English phonemes (24 consonants, 15 vowels) and 58 pinyin symbols (23 initials and 35 finals), there are always some indicative elements, i.e. *indicators*, which facilitate alignment. For E, they are:

- all the consonants;
- vowel at the first position;
- and the second vowel of two contiguous vowels.

Correspondingly in C, they are:

- all the initials;
- final at the first position;
- and the second final of two contiguous finals.

Note that similar indicators can be easily identified in other Chinese Romanization systems. Hence, they are independent of alignment model. Also, we define the following variables:

- $\tau(S) = \#$ *of indicators in sequence* S, $S \in \{E, C\}$
- $t(E, C) = \max\{\tau(E), \tau(C)\}$, represents the maximum number of indicators in E and C.
- $d(E, C) = |\tau(E) - \tau(C)|$, is the difference of the number of indicators in E and C.

We chunk E and C by tagging the identified indicators and compensate the one with fewer indicators by inserting d number of mute ε at its $min\{\tau(E), \tau(C)\}$ possible positions ahead of its indicators. ε is practically an indicator defined for alignment. This ensures that both sequences end up with the same number of indicators. The t chunks separated by indicators in E should align to the corresponding t chunks in C in the same order. They are called *alignment chunks*. There are $\|A\| = \begin{pmatrix} d \\ t \end{pmatrix} = \frac{t!}{(t-d)!d!}$ number of possible alignments at chunk-level with respect to different positions of ε.

This method can assure that each chunk contains two sound units at most. Thus, in a pair of aligned chunks, only three mapping layouts are possible for individual phoneme elements, i.e. individual consonants, vowels, initials and finals:

1. e-*to*-c_1c_2: The alignment at phoneme level would be kept as e-*to*-c_1c_2, where c_1c_2 is considered as an initial-final *cluster*;
2. e_1e_2-*to*-$c1c_2$: The alignment at phoneme level would be extended to e_1-*to*-c_1 and e_2-*to*-c_2. Note that this condition will not generate new alignment. Thus, the overall number of alignment remains unchanged.
3. e_1e_2-*to*-c: By adding an additional ε at C side, the alignment at phoneme level would be extended to e_1-*to*-c and e_2-*to*-ε or e_1-*to*-ε and e_2-*to*-c. In this case, one more new alignment will be produced and we update $\|A\| = \|A\| + 1$.

Figure 4.2 shows an example of the alignment chunks (indicators are tagged by '|') between /AE L B AH K ER K IY/ (Albuquerque) and /a er bo ke er ji/ (阿尔伯克尔基). Our chunk-based alignment scheme works as follows: The English pronunciation should be first compensated by inserting one ε as it has one fewer indicator than its Chinese counterpart. There are 6 possible alignments at chunk level corresponding to the 6 possible positions for the inserted ε. However, the total possible alignments at phoneme level would be 11 because of the existence of /K ER/-to-/er/ in the first four chunk-level alignments and /K IY/-to-/er/ in the sixth chunk-level alignment.

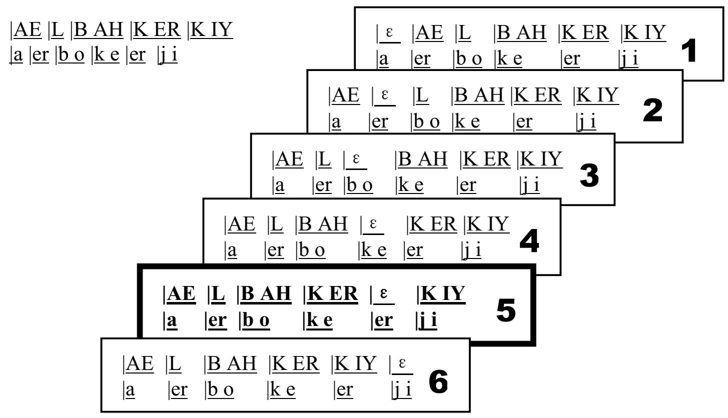
|AE |L |B AH |K ER |K IY
|a |er |b o |k e |er |j i
|ε |AE |L |B AH |K ER |K IY
|a |er |b o |k e |er |j i
1
|AE |ε |L |B AH |K ER |K IY
|a |er |b o |k e |er |j i
2
|AE |L |ε |B AH |K ER |K IY
|a |er |b o |k e |er |j i
3
|AE |L |B AH |ε |K ER |K IY
|a |er |b o |k e |er |j i
4
|AE |L |B AH |K ER |ε |K IY
|a |er |b o |k e |er |j i
5
|AE |L |B AH |K ER |K IY |ε
|a |er |b o |k e |er |j i
6

Figure 4.2: An example depicting the alignment of phoneme chunks

4.3 Transliteration Model Training

4.3.1 *EM* Training for Symbol-mappings

We then apply *EM* algorithm [19] to find the *Viterbi* alignment (the most probable alignment) for each training pair. The training process is described in the Algorithm 1.

Algorithm 1 *EM* training algorithm for *Viterbi* alignment and symbol-mapping probabilities

1: **Initialization:** For each English-Chinese pair, assign equal weights to all alignments generated based on phoneme chunks as $||A||^{-1}$.

2: **Expectation:** For each of the 39 English phonemes, count the instances of its different mappings from the observations on all alignments produced. Each alignment contributes counts in proportion to its own weight. Normalize the scores of the mapping units it maps to so that the mapping probability sums to 1.

3: **Maximization:** Re-compute the alignment scores. Each alignment is scored with the product of the scores of the symbol mappings it contains. Normalize the alignment scores so that each pair's alignment scores sum to 1.

4: **Repeat:** Repeat step 2-3 until the symbol-mapping probabilities converge, meaning that the variation of each probability between two iterations becomes less than a specified threshold.

The direct transliteration model $Pr(C|E)$ is estimated by *EM* training

Foreign Name	Chinese Transliteration	*Viterbi* Alignment Found							Weight
Cirelli	奇雷利	SH	IH	R	EH	L	IY		1.000
		qi	l	ei	l	i			
Colyer	科利尔	K	OW	L	IY	ER			1.000
		k	e	l	i	er			
Brag	布拉格	B	R	AE	G				1.000
		bu	l	a	ge				
Karami	卡拉米	K	ER	AA	M	IY			1.000
		k	a	la	m	i			
Deloris	德洛里	D	EH	L	ER	IH	S		1.000
		d	e	l	uo	li	ε		
Schoenwald	舍恩瓦尔德	SH	OW	N	W	AO	L	D	0.993
		sh	e	en	w	a	er	de	
Hausner	奥斯内	HH	AW	S	N	ER			1.000
		ao	ε	si	n	ei			
Hudecek	胡德切克	HH	AH	D	IH	CH	EH	K	1.000
		h	u	d	e	q	ie	ke	
Brearley	布里尔利	B	R	ER	ε	L	IY		0.998
		bu	l	i	er	l	i		
Anatole	阿纳托尔	AE	N	AH	T	OW	L		1.000
		a	n	a	t	uo	er		
...	...	...							...

Table 4.1: Sample *Viterbi* alignments learned by *EM* Training

on the data set *Base-0* with 41,674 parallel instances. Compared to the brutal force alignment computation [19], our *EM* training based on alignment of phoneme chunks produces significantly fewer possible alignments. Thus fewer possible symbol-mappings for each English phoneme are involved. Mappings crossing chunks are also avoided. Therefore these symbol-mappings tend to be more accurate. For each English-Chinese pair, the *Viterbi* alignment is found whose alignment score (weight) approaches to 1 with the increase of iteration times. Table 4.1 shows the randomly selected sample *Viterbi* alignments for each name pair through automatic *EM* training. Table 4.2 shows the top-4 pinyin symbol-mapping probabilities for each English phoneme after *EM* converging at a predefined threshold of $1.0E-4$.

The mute ε is introduced to both sides of phonetic alphabets during the processing of phoneme chunks. It plays an important role for carrying out "virtual mapping" with respect to the conditions 3 and 4 (see Section 4.1). The *EM* training initiated with alignment based on phoneme chunks automatically calculates the mapping probabilities from each English phoneme to not only individual initials and finals, but also to initial-final *clusters*. From the training instances, the algorithm identifies these clusters, e.g. /d-e/, /k-e/, /s-i/, /t-e/, etc. They are dynamically appended in pinyin inventory as additional candidates for transcriptions from English phonemes like /D/,

e	c	$p(c\|e)$	c	$p(c\|e)$	c	$p(c\|e)$	c	$p(c\|e)$
AA	a	0.61923	uo	0.09505	e	0.07352	o	0.06314
AE	a	0.86606	ia	0.03544	ε	0.02509	ai	0.01867
AH	a	0.39206	e	0.19805	u	0.10216	i	0.07872
AO	uo	0.25501	o	0.17831	ao	0.15931	e	0.15428
AW	ao	0.55706	uo	0.11303	u	0.09228	a	0.08996
AY	ai	0.51124	i	0.22925	ei	0.09985	a	0.06756
B	b	0.75535	bu	0.20321	w	0.00789	bi	0.00786
CH	q	0.42866	qi	0.15122	ch	0.11413	x	0.05970
D	d	0.61585	de	0.29259	ε	0.06035	er	0.00629
DH	s	0.46763	t	0.20863	x	0.07914	si	0.07194
EH	ei	0.32765	e	0.22758	ai	0.22453	a	0.05100
ER	e	0.38604	ei	0.14128	a	0.10870	o	0.10745
EY	a	0.38817	ai	0.22236	ei	0.20053	e	0.05488
F	f	0.60277	fu	0.37734	fei	0.00766	p	0.00227
G	g	0.39740	ge	0.29644	j	0.17965	ε	0.04828
HH	h	0.65573	x	0.08640	a	0.06690	ai	0.04911
IH	i	0.70619	ei	0.11094	e	0.05979	e	0.03180
IY	i	0.70956	ei	0.11083	ai	0.05104	e	0.03432
JH	j	0.31634	y	0.13153	g	0.10053	qi	0.07931
K	k	0.43570	ke	0.31736	j	0.11575	ε	0.02968
L	l	0.60161	er	0.34286	ε	0.01379	le	0.01196
M	m	0.77637	mu	0.10848	n	0.05647	ng	0.03167
N	n	0.79253	ng	0.16759	nei	0.01292	na	0.00682
NG	n	0.55059	ng	0.33577	g	0.07692	r	0.01033
OW	uo	0.45049	e	0.19401	o	0.13831	ao	0.08572
OY	uo	0.33626	u	0.21314	o	0.20175	e	0.08224
P	p	0.54823	pu	0.25763	b	0.16020	pei	0.01624
R	l	0.69558	er	0.14198	ϵ	0.13104	e	0.00816
S	si	0.41800	s	0.26173	x	0.08858	shi	0.07538
SH	shi	0.33762	sh	0.32755	x	0.22805	q	0.01116
T	t	0.38772	te	0.30403	d	0.14454	ε	0.11809
TH	si	0.27598	s	0.19066	te	0.18676	t	0.16177
UH	u	0.71079	i	0.06220	o	0.05288	ε	0.02813
UW	u	0.57044	ε	0.10436	i	0.05350	iu	0.04428
V	w	0.66063	f	0.12757	fu	0.10999	b	0.07295
W	w	0.77548	h	0.06043	k	0.04924	a	0.02539
Y	y	0.37204	ε	0.19440	h	0.05758	w	0.04427
Z	si	0.44383	s	0.08952	z	0.06873	ci	0.06819
ZH	r	0.25000	j	0.17391	x	0.11778	y	0.09799
ε	er	0.33699	yi	0.13621	n	0.11528	e	0.08478

Table 4.2: English phonemes with probabilistic mappings to Chinese pinyin sound units.

/K/, /S/, /T/, etc. This can reduce the error due to zero-fertility symbols in the source-channel model.

4.3.2 *WFST* for Phonetic Transition

We then build a *weighted finite state transducer* (WFST) using AT&T FSM library[1] based on the symbol-mapping probabilities in Table 4.2 for the transcription of an input English phoneme sequence into its possible pinyin symbol sequences. Each arc carries the transition information from an English phoneme to its pinyin counterparts as well as their transition cost, which is given by $1 - p(c|e)$. Figure 4.3 shows part of the transducer, including the transitions for several English phonemes. There are about 2,666 arcs included in the actual automata. Note that arcs like [/AA/:/uo/|0.904], whose output symbol includes multiple characters, are split into multiple arcs, i.e. [/AA/:/u/|0.904] and [/ε/:/o/|0.0] jointed by an intermediate nodes like nodes $1, 2, \ldots, 5, \ldots$. This is for the following pinyin syllable transducer for pinyin syllable segmentation being able to connect with it.

4.3.3 Issues for Incorrect Syllables

Many of the pinyin symbol sequences produced by the transliteration model WFST cannot be correctly syllabified or include illegitimate pinyin syllables as the transducer itself has no knowledge about pinyin's regulations. Actually only 396 of $23 * 25$ possible combinations of initials and finials can constitute legal pinyin syllables. Legal syllables, regardless of dialects, can be easily collected from Chinese lexicons in corresponding Romanization systems by using an automatic scanning program. We automatically collect these legal syllables from the pinyin part of *Base-0*. Based on this knowledge, we construct a *finite state transducer* (FST) with about 1,284 arcs. Figure 4.4 presents only several syllables of this transducer. The composition between the FST and the previous WFST can eliminate illegal pinyin symbol sequences and segmenting legal sequences into syllables.

4.4 Language Model Training

A syllable-based bigram language model of pinyin is trained using the Chinese part of the same 41,674 training instances in *Base-0*, on which the transliteration WFST was built. The model $\Pr(C)$ is approximated by counting

[1]http://www.research.att.com/~mohri/fsm/

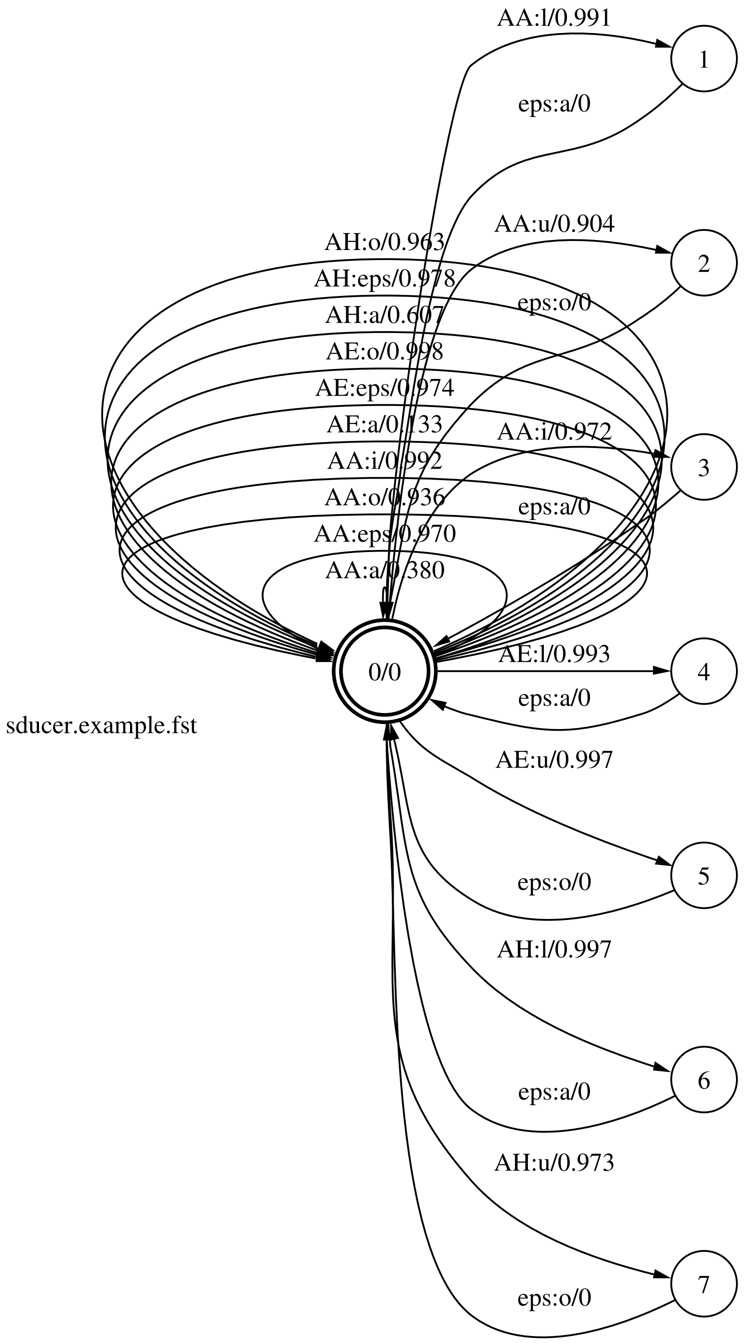

Figure 4.3: Part of the WFST based on symbol-mapping probabilities $p(c|e)$. *eps* denotes ε

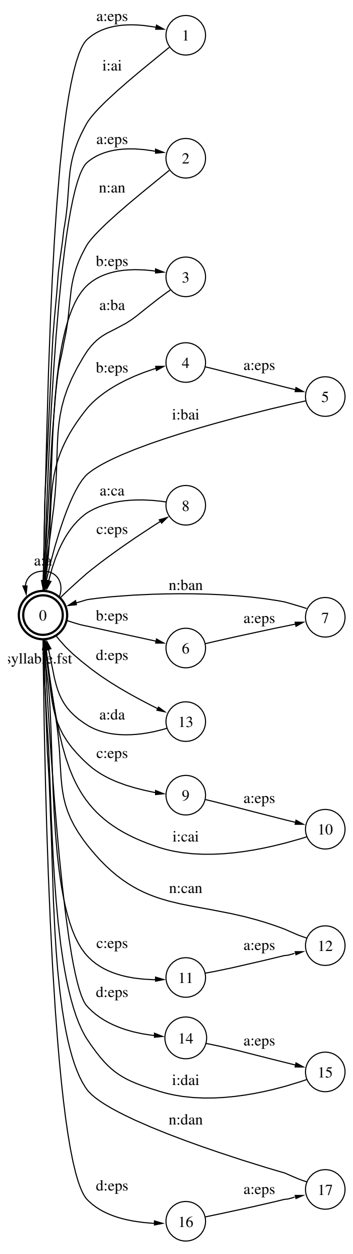

Figure 4.4: Part of the FST for pinyin syllabification. *eps* denotes ε

the frequencies of syllable occurrences in this data set using the following equation:

$$\Pr(C) \simeq \prod_i p(c_i|c_{i-1}) = \prod_i \frac{count(c_{i-1}c_i)}{count(c_i)} \tag{4.4}$$

where c_i is the pinyin syllable of a Chinese character.

We then implement the bigram model using a *weighted finite state acceptor* (WFSA) with one state for each item in the pinyin syllable vocabulary. Between each pair of states, say x/y, there is a single transition whose label is the syllable y and whose probability is $p(y|x)$. We then add a special final state with transitions leading to it from every other state labeled by ε with probability 1.0. Finally, a start state is added with transitions to every state y with label y and probability $p(y)$. The WFSA is used to re-rank the pinyin syllable sequences yielded by the k-best path algorithm from the composition of the previous two transducers. The WFSA is partially shown in Figure 4.5 where only ten states are presented. The actual automata includes 330 states and 12,274 arcs corresponding to the syllables and bigram dependencies in training corpus. Since we don't smooth the unobserved syllables in the data, the number of states are less than that of all possible pinyin syllables.

4.5 Search Algorithm

Given an input English phoneme sequence, it is first built as an input *finite state acceptor* (FSA). Searching could be conducted by successive compositions of the input FSA with the three automatas from training process using the composition algorithms provided by AT&T FSM toolkits [24]. However, if the language model is applied to the entire space of syllables generated by the previous two transducers, we could hardly obtain correct transliterations because short hypotheses were ranked high by the k-shortest path algorithm [11]. One adjustment is adopted by applying bigram search to only the first m candidates produced by the transducers for each given English name. m is empirically set as 250 (see Section 4.6.4). The function of the language model is to re-rank the m candidates according to bigram dependencies. Although this may cause possible loss of optimal hypotheses, it can significantly reduce searching errors. Another reason to use only m candidates is that the search space of the syllables generated by the transducers is extremely large, which renders the search process very time-consuming. We then output the top-n transliterations from those re-ranked m candidates according to their transition probabilities from the transducers and bigram probabilities from the acceptor. Figure 4.6 illustrates the top-10 transliterations given the input

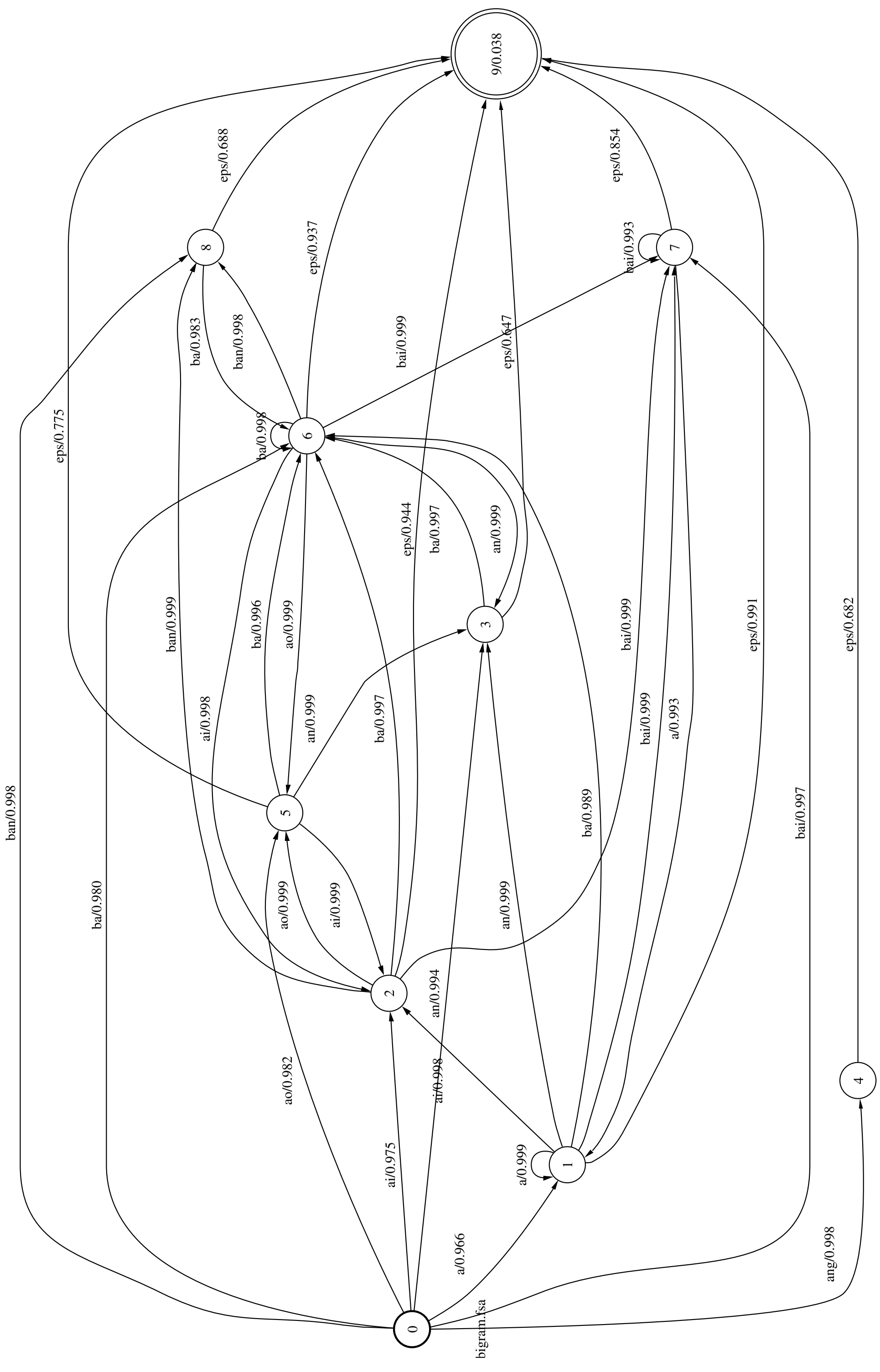

Figure 4.5: Part of the WFSA for the bigram language model on pinyin syllables. *eps* denotes ε

$C.A.$ Range(%)	[0 $\sim$ 20)	[20 $\sim$ 40)	[40 $\sim$ 60)	[60 $\sim$ 80)	[80 $\sim$ 100)	[100]
Close	4.30%	9.22%	27.22%	34.66%	11.24%	13.36%
Open	4.15%	9.86%	28.05%	33.50%	13.10%	11.34%

Table 4.3: $C.A.$ distribution results of Experiment I

name /K EH V IN N/ (Kevin).

4.6 Experimental Results

To evaluate the transliteration performance of **Direct-1** more comprehensively, the following experiments were conducted on data set *Base-0*, which included but were not limited to the comparisons with the baseline.

4.6.1 Experiment I: $C.A.$ Distribution

In this experiment, only top-1 machine transliteration of each name was chosen for comparison with the standard transliteration. A name often has multiple transliteration alternatives. Hence, we measured how the percentage of the number of generated transliterations distributes over different character-level accuracy ranges, which is referred to as $C.A.$ Distribution:

$$C.A.D.(r1, r2) = \frac{\#\ of\ names\ with\ C.A. \in [r1, r2)}{\#\ of\ tested\ names} \tag{4.5}$$

where [r1,r2) is the bound of a $C.A.$ range. We set up six $C.A.$ ranges: [0% $\sim$ 20%)[2], [20% $\sim$ 40%), [40% $\sim$ 60%), [60% $\sim$ 80%), [80% $\sim$ 100%) and [100%]. We are particularly interested in the names within the $C.A.$ ranges of [0% $\sim$ 20%) and [80% $\sim$ 100%) since the former could be considered as "completely incorrect" while the latter "acceptable".

We counted the number of names whose $C.A.$ fall in each range. The percentages are listed in Table 4.3.

4.6.2 Experiment II: Top-n Accuracy

We collected the top-50 transliteration results for each foreign name. We counted the number of correct ones in the resulting top-n ($n \in \{10, 20, 30, 40, 50\}$) transliterations whose $C.A.$ is 100% for a given name. The percentages are listed in Table 4.4. This experiment evaluated the proportion of instances whose correct transliteration could be found in top-n generated results.

[2]'[M%∼N%)' denotes $\geq M\%and < N\%$.

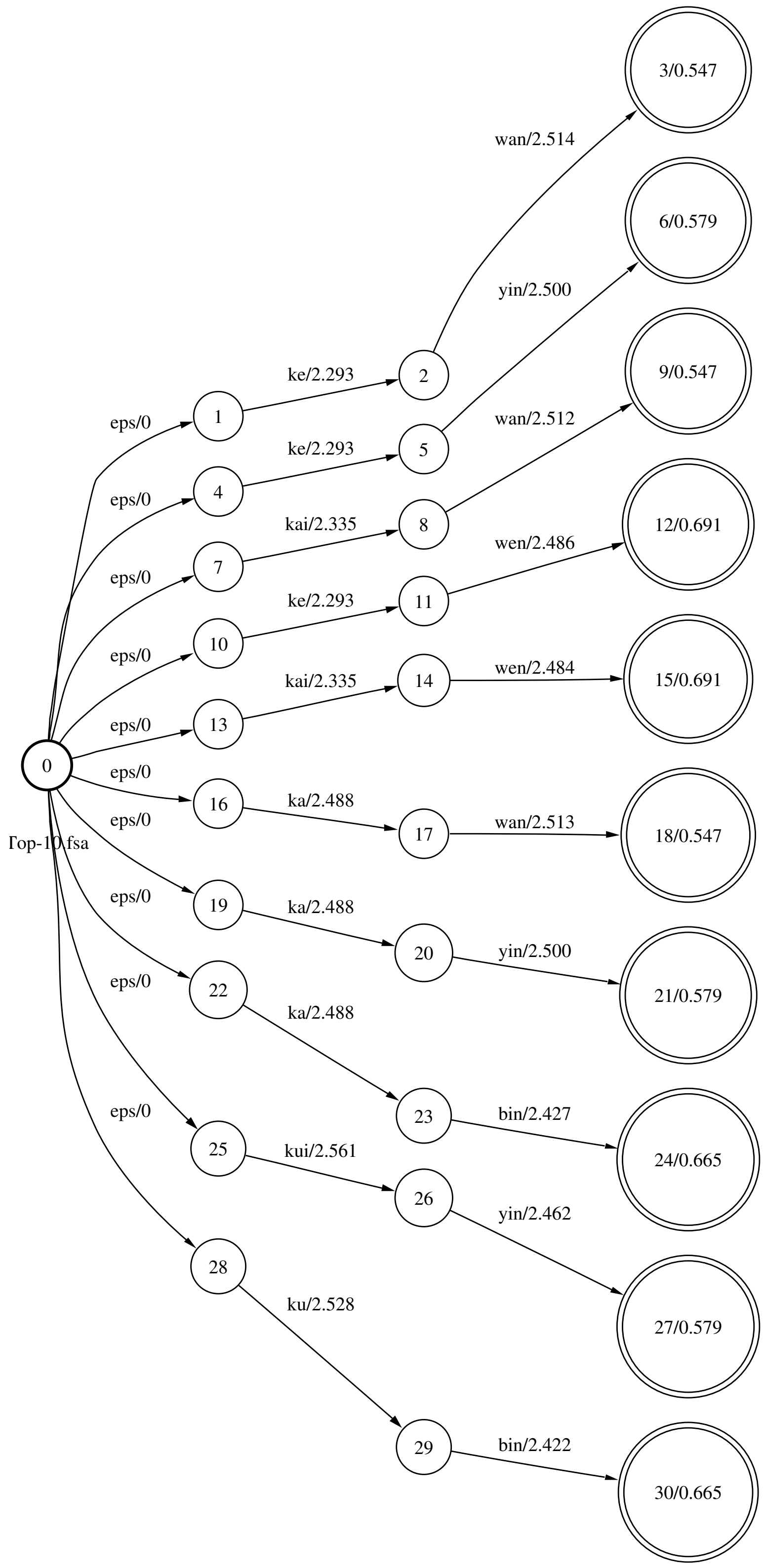

Figure 4.6: Sample output of top-10 transliterations given the name "Kevin"

Top n	1	10	20	30	40	50
Close	13.36%	51.13%	57.29%	58.83%	59.35%	59.52%
Open	11.34%	46.99%	53.01%	54.55%	54.96%	55.12%

Table 4.4: The percentage of correct transliterations found in top-n results

Systems		Baseline	Direct-1
C.A.	Close	66.35%	63.17%
	Open	65.15%	62.61%
W.A.	Close	20.73%	13.36%
	Open	18.27%	11.34%

Table 4.5: Transliteration accuracies compared to the source-channel based system

4.6.3 Experiment III: Comparisons with the Baseline

In this experiment, we evaluated our direct transliteration model with *C.A.* and *W.A.* measurements (see Section 3.3.2) on data set *Base-0* for comparisons with our implementation of the source-channel baseline. Only top-1 machine transliteration was used. The results are shown in Table 4.5.

4.6.4 Experiment IV: Influence of m Candidates

To examine the suitable m value of candidates used for searching, we made the transliteration transducers generate top-m ($m = 200, 250, 500, 750, 1000, 10000$) candidates and apply the bigram WFSA to them to find the top-1 transliteration in the open test. The *C.A.* value changed with the different m values and are shown in Figure 4.7.

4.7 Discussions

In experiment I (see Table 4.3), the possibility of finding correct transliterations in top-1 result candidates was fairly low. Only 13.36% and 11.34% test instances were correct. If we considered “acceptable” transliterations as *C.A.* greater than 80%, the accumulative percentages would be 24.60% and 24.44% for close and open tests respectively. Two pinyin sequences having 20% edit distance can be exemplified as /ben la deng/ (本拉登) and /ben la dan/ (本拉丹). Because the average length of testing names is generally about 5 to 6 phonemes, machine transliterations with *C.A.* value ranging in

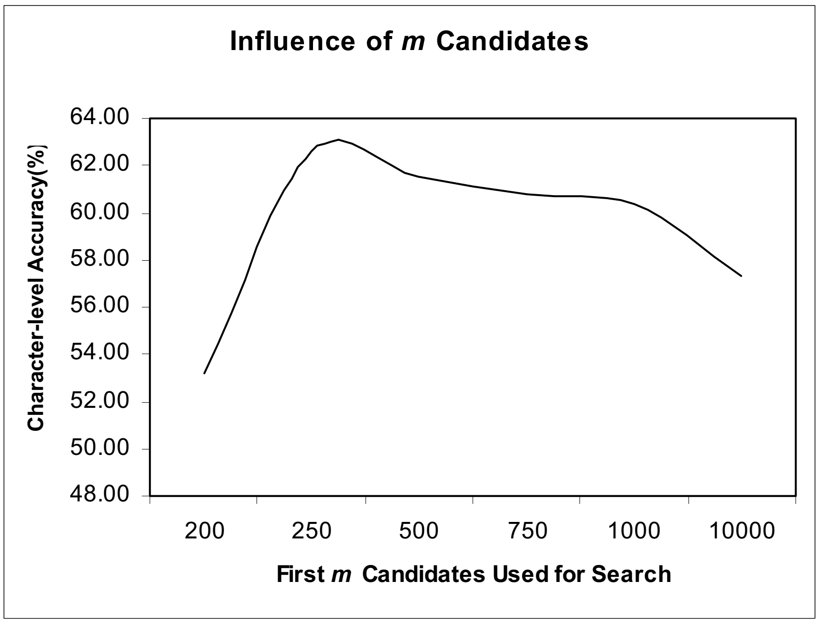

Figure 4.7: Influence of using the first m candidates for search

[80% $\sim$ 100%) basically imply that 1 (or less) of 5 or 6 phonemes are mismatched with the standard. Hence they can be considered as phonetically equivalent but misspelled, which is a measurement used in human subject tests [2, 19].

In experiment II (see Table 4.4) where top-50 transliterations were examined. 51.13% names in close test and 46.99% in open test had their correct transliterations within the top-10 results; and 59.52% names in close test and 55.12% in open test had correct transliterations within top-50 results. We also observed considerable increase in the percentage of correct transliterations if we compared top-20 results with only top-1. But no apparent improvement was achieved if we considered more transliteration results, e.g. from top-20 to top-50.

In experiment III (see Table 4.5), our approach demonstrates comparable *C.A.* value to that of the baseline, and is slightly worse by 3.18% on close test and 2.54% on open test. However, the *W.A.* measurement shows that the baseline performed about 7.37% and 6.93% better than **Direct-1** for close and open tests respectively. The lower accuracy of our direct model results from two main reasons:

1. Although the direct model in Eq.(4.1) is mathematically correct under the MaxEnt framework, the accuracy of the posterior estimation de-

pends on f_m—the feature functions selected, and the parameters λ_m. Recall that we selected the log function features of the two models and set parameters $\lambda_m = 1$ for the soundness of Eq.(4.1). This assumes equal contribution of the two models. Hence, the approximation is worse than the source-channel model, whose maximization is supported by Bayes' rule. Improvements can be made by manually adjust the weight on the two models.

2. Our search algorithm used a rough approximation, which can be further improved. Because the search algorithms provided by automata are based on shortest-path, short candidates produced by the first two transducers would be ranked high by the bigram acceptor. Our method makes a comprise by only using a limited number of candidates generated by the first two transducers. This could reduce the chances for shorter candidates in some certain range, but may exclude the correct hypotheses that are ranked lower than value m by the transducers. As m increases, it is inevitable that short names are produced and the comprise tends to be useless, evidenced as the illustration of Figure 4.7 in the experiment IV. Improvements could be made by raising the cost of the candidates, which are obviously shorter than the given testing name, and by increasing the m value to broaden the search scope at the same time.

We will leave the improvements over thse two shortcomings for future research. The proposed **Direct-1** model demonstrate the feasibility of a simple direct statistical transliteration method. This simple method is comparable to the approach based on sophisticated techniques on *C.A.* measurement and a bit worse on *W.A.* mainly due to the search techniques used. Also, the direct approach can overcome **Problem 1** and **Problem 2** in IBM SMT based system. This is achieved by adopting a one-to-many mapping from source to target phonetic symbols. More importantly, the advantage of direct method lies in its flexibility to further incorporate useful features based on dependencies among surrounding phonemes. The neighboring pronunciation units being transcribed can provide significant information for determining their mapping probabilities. For example, both /*AH* P AA L OW/ (Appollo), which is translated as /a bo luo/ (阿波罗), and /W AO L *AH* S/ (Wallace), which is translated into /hua lai shi/ (华莱士), both contain the sound /AH/, but /AH/ are mapped to different pinyin sounds, i.e. /a/ and /ai/, respectively in different context.

4.8 Chapter Summary

In this chapter, we modeled the statistical transliteration problem as a direct phonetic symbol transcription model plus a language model for post-adjustment. Although the performance of the proposed direct method is lower than the source-channel based system, it overcomes **Problem 1** and **Problem 2** of the baseline. Also, the advantage of direct method is its flexibility for incorporating features based on dependencies among surrounding phonemes. In the next chapter, we will further propose a direct transliteration model. The enhanced model will make use of contextual feature functions within MaxEnt framework [4].

□ **End of chapter.**

Chapter 5

Improving Direct Transliteration

The limitation of **Problem 3** (see Section 3.2) shows that it is difficult to expand IBM SMT to consider flexible context information. And **Problem 4** suggests that leveraging separate transliteration model and language model to achieve the best results out of their combination is difficult. This motivates us to propose an improved direct transliteration method to overcome these problems. This new model is referred to as **Direct-2** in this thesis.

General Idea: According to Eq.(4.3), we can arbitrarily and flexibly choose feature functions f_m. Thus, the language model can be considered as an optional feature under MaxEnt framework [26]. Unfortunately, there is no effective feature selection techniques available that could combine features for the language model with that of the direct transliteration model. However, the direct transliteration model would work well without the language model if other cutting edge features were chosen.

5.1 Improved Direct Model—Direct-2

5.1.1 Enlightenment from Source-Channel

We re-examined the transliteration portion source-channel model in Eq.(2.4) and reversed the order of the source and target. According to [4, 7], the translation model in source-channel is given by the following formula:

$$\Pr(C|E) = \sum_{A} \Pr(C, A|E) \approx p(C, \hat{A}|E) \tag{5.1}$$

where A denotes the hidden parameter for alignment between E and C, and $\hat{A}$ is assumed to be the *Viterbi* (the most probable) alignment. The approximation is sound because the translation probability with respect to the *Viterbi* alignment can dominate the summation of the probabilities of all the possible alignments. $p(C, \hat{A}|E)$ can be derived from the "basic translation model" in [4]:

$$p(C, \hat{A}|E) = \prod_{i=1}^{|E|} p(n(e_i)|e_i) \times \prod_{j=1}^{|C|} p(c_j|e_{a_j}) \times d(\hat{A}|E, C) \tag{5.2}$$

In this expression,

- $p(n(e_i)|e_i)$ is the probability that the English phoneme e_i generates $n(e_i)$ number of pinyin symbols, i.e. the fertility of e_i;
- $p(c_j|e_{a_j})$ is the probability that the English phoneme e_{a_j} generates the pinyin symbol c_j. For every pinyin symbol position j in C, a_j is the phoneme position in E of the English phoneme that maps to c_j in the given alignment $\hat{A}$.
- $d(\hat{A}|E, C)$ is the probability of the particular order of pinyin symbols, i.e. the distortion probability when the target symbols are generated.

We first note that the distortion probability, which is originally a parameter in machine translation taking care of particular order of the target words, is unnecessary in our task since the order of generated pinyin symbols strictly follows the order of the source English phonemes. Thus it can be dropped.

Also, we simplified $p(c_j|e_{a_j})$. To reduce parameters, we introduce the Chinese pinyin *mapping units* (*cmu*) of each e_i denoted by cmu_i, which can be individual pinyin symbols or *clusters* of initials and finals. In an alignment, each English phoneme aligns to only one *cmu*. Thus, the parameter $p(n(e_i)|e_i)$ can also be removed. Then $p(C, \hat{A}|E)$ is approximated by:

$$p(C, \hat{A}|E) \approx \prod_{i=1}^{|E|} p(cmu_i|e_i) \tag{5.3}$$

The unknown *cmus* (initials, finals, or *clusters*) can be discovered "on the fly" during *EM* training for computing the *Viterbi* alignments and symbol-mapping probabilities according to Section 4.3. In fact, they can also be obtained by the *EM* algorithm in GIZA++ by reversing the order of E and C in the source-channel transliteration model training.

5.1.2 Using Contextual Features

Eq.(5.3) gives poor approximation as no contextual feature is considered. Based on the discussions in the previous chapter (see Section 4.7), we consider transliteration as a classification problem, which is to classify each phoneme of a given English name into its most probable *cmu* according to the frames of various constraints including its neighboring phonemes, the targets of its neighboring phonemes, or even other individual models like language model [26]. This leads to better approximation:

$$p(C, \hat{A}|E) \simeq \prod_{i=1}^{|E|} p(cmu_i|h_i) \tag{5.4}$$

where h_i is the *history* or *context* of e_i, which can be defined as follows:

$$h_i = \{e_i, e_{i+1}, e_{i+2}, e_{i-1}, e_{i-2}, cmu_{i-1}, cmu_{i-2}\} \tag{5.5}$$

History of an English phoneme is defined as its left-two and right-two neighboring phonemes plus the two *cmu*s at pinyin side, to which its left-two phonemes align.

5.1.3 Estimation Based on MaxEnt

For each e in a given pair of $\{e_1, e_2, \ldots, e_n\}$ and $\{cmu_1, cmu_2, \ldots, cmu_n\}$, its conditional transliteration probability to produce *cmu* with respect to its contextual history h can be computed by

$$p(cmu|h) = \frac{p(h, cmu)}{\sum_{cmu' \in \Omega} p(h, cmu')} \tag{5.6}$$

where Ω is the set of all *cmus* mapped from e observed in the training data, and $p(h, cmu)$ is the joint probability distribution of observing h and *cmu* simultaneously. $p(h, cmu)$ can be trained using maximum likelihood estimation, i.e. to find the model $p_\lambda(h, cmu)$ that maximizes the likelihood of the training data:

$$p_\lambda = \operatorname*{argmax}_p L(p)$$

where λ is the model parameter.

By introducing a set of "features" $\{f_1, f_2, \ldots, f_m\}$ and their corresponding parameters $\lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_m\}$ to express observed events, say (h, cmu), in the data, this model also can be obtained under the well-established *MaxEnt* formalism, in which the goal of the model is to maximize the *entropy* of the distribution under certain constraints [4]:

$$p_\lambda = \operatorname*{argmax}_p H(p)$$

where

$$H(p) \equiv - \sum_{h,cmu} \{p(h, cmu) \log p(h, cmu)\}$$

and constraints are given by

$$E(f_i) = \tilde{E}(f_i), 1 \leq i \leq m \tag{5.7}$$

$E(f_i)$ is the feature expectation of the model defined and approximated as [30]:

$$\begin{aligned} E(f_i) &= \sum_{h,cmu} \{p(h, cmu) f_i(h, cmu)\} \\ &\simeq \sum_{j=1}^{n} \{\tilde{p}(h_j) p(cmu_j|h_j) f_i(h_j, cmu_j)\} \end{aligned}$$

where $\tilde{p}(h_j)$ is the probability of observed history h_j in the training data. $\tilde{E}(f_i)$ is the feature expectation of the empirical distribution obtained from training data:

$$\tilde{E}(f_i) = \sum_{j=1}^{n} \{\tilde{p}(h_j, cmu_j) f_i(h_j, cmu_j)\}$$

where $p(h_j, cmu_j)$ denotes the observed probability of (h_j, cmu_j) in the training data.

This two expectations are forced to be equivalent in Eq. (5.9) under the restriction that the inferences from the model should match with observations from the real data. This constrained optimization problem is to find the model that has the form [10, 30]:

$$p(h, cmu) = \mu \prod_{j=1}^{m} \lambda_j^{f_j(h,cmu)} \tag{5.8}$$

where $\{\mu, \lambda_1, \lambda_2, \ldots, \lambda_m\}$ are the model parameters and $\{f_1, f_2, \ldots, f_m\}$ are binary-valued features functions. Each parameter λ_j corresponds to a feature f_j.

In general, we have the following theorem and its proof can be found in [10]:

Theorem 1 *Let I be a finite set and $p = \{p_i; i \in I, p_i \geq 0, \sum_{i \in I} p_i = 1\}$ be a probability function on I. Let*

$$\sum_{i \in I} b_{si} p_i = k_s, s = 1, 2, \ldots, d, \tag{5.9}$$

be the constraints, where $\forall s, \exists i \in I$ such that $b_{si} \neq 0$, and b_{si} is given, and μ, μ_s are to be found. If there exists a positive probability function of the form

$$p_i = \mu \prod_{s=1}^{d} \mu_s^{b_{si}}$$

satisfying the constraint of Eq.(5.9), then it maximizes the entropy

$$H(p) = -\sum_{i \in I} p_i \log p_i$$

and is unique in doing so.

It is shown that if p has the form Eq.(5.8) and satisfied constraint Eq.(5.7), it uniquely maximizes the entropy $H(p)$ over distribution p. The model parameters for the distribution $p(cmu, h)$ can be obtained via the *Generalized Iterative Scaling* (GIS) algorithm [10, 30].

5.1.4 Features for Transliteration

In Figure 4.1, for example, a binary feature can be identified from training corpus taking the following form:

$$f_1(h_i, cmu_i) = \begin{cases} 1 & \text{if } e_i\text{=/F/ and } e_{i-1}\text{=START and } e_{i+1}\text{=/R/ and} \\ & cmu_i\text{=/fu/} \\ 0 & \text{otherwise} \end{cases}$$

or,

$$f_2(h_i, cmu_i) = \begin{cases} 1 & \text{if } e_i\text{=/S/ and } e_{i+1}\text{=/T/ and } cmu_{i-1}\text{=/IH/ and} \\ & cmu_i\text{=/si/} \\ 0 & \text{otherwise} \end{cases}$$

Take the first feature above for example: if the feature exists in the feature set defined in the model, its corresponding model parameter λ_i will contribute towards the joint probability $p(h_i, cmu_i)$ when e_i starts with /F/ followed by /R/ and its related cmu_i is /fu/ (see Figure 4.1).

The feature set can be empirically defined according to specific applications. Theoretically, a feature can be generated from any possible contextual knowledge without restrictions. However, considering the computational complexity, the scope of the history usually greatly reduces to a relatively practical range in practice. The general feature set we used in experiments are listed in Table 5.1. It acts as the templates used for extracting features from training corpus. $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ are called instantiations variables, which are instantiated automatically by the corresponding English phonemes and pinyin *cmu*s from the training set. $|\mathcal{V}_{\mathcal{E}}|$ and $|\mathcal{V}_{\mathcal{C}}|$ are the sizes of English phoneme vocabulary and pinyin *cmu* vocabulary, respectively.

For example, given an aligned pair of phoneme-pinyin sequences in Table 5.2 and suppose the current English phoneme is e_7, the features with respect to its context h_7 and the prediction cmu_7 can be extracted form the data, which are shown in Table 5.3.

Category	Contextual Feature Templates	# of Possible Features
1	$e_i = \mathcal{X}$ and $cmu_i = \mathcal{Z}$	$\lvert\mathcal{V}_\mathcal{E}\rvert \cdot \lvert\mathcal{V}_\mathcal{C}\rvert$
2	$cmu_{i-1} = \mathcal{X}$ and $cmu_i = \mathcal{Z}$	$\lvert\mathcal{V}_\mathcal{C}\rvert^2$
3	$cmu_{i-2}cmu_{i-1} = \mathcal{X}\mathcal{Y}$ and $cmu_i = \mathcal{Z}$	$\lvert\mathcal{V}_\mathcal{C}\rvert^3$
4	$e_{i-1} = \mathcal{X}$ and $cmu_i = \mathcal{Z}$	$\lvert\mathcal{V}_\mathcal{E}\rvert \cdot \lvert\mathcal{V}_\mathcal{C}\rvert$
5	$e_{i-2} = \mathcal{X}$ and $cmu_i = \mathcal{Z}$	$\lvert\mathcal{V}_\mathcal{E}\rvert \cdot \lvert\mathcal{V}_\mathcal{C}\rvert$
6	$e_{i+1} = \mathcal{X}$ and $cmu_i = \mathcal{Z}$	$\lvert\mathcal{V}_\mathcal{E}\rvert \cdot \lvert\mathcal{V}_\mathcal{C}\rvert$
7	$e_{i+2} = \mathcal{X}$ and $cmu_i = \mathcal{Z}$	$\lvert\mathcal{V}_\mathcal{E}\rvert \cdot \lvert\mathcal{V}_\mathcal{C}\rvert$

Table 5.1: Contextual feature templates in improved model **Direct-2**

Position	1	2	3	4	5	6	7	8	9	10	11
English	F	R	AE	N	S	IH	S	T	EY	L	ER
Chinese	fu	l	ang	ε	x	i	si	t	ai	l	e

Table 5.2: A given alignment in training data

	Feature Contexts	Feature Predictions
$feature_1$:	e_i =/S/	and cmu_i =/si/
$feature_2$:	e_{i-1} =/IH/	and cmu_i =/si/
$feature_3$:	cmu_{i-1} =/i/	and cmu_i =/si/
$feature_4$:	e_{i-2} =/S/	and cmu_i =/si/
$feature_5$:	cmu_{i-2} =/x/ and cmu_{i-1} =/i/	and cmu_i =/si/
$feature_6$:	e_{i+1} =/T/	and cmu_i =/si/
$feature_7$:	e_{i+2} =/EY/	and cmu_i =/si/

Table 5.3: Features extracted from h_7 =/S/ for predicting cmu_7=/si/

Training Size	cut-off	# of contexts	# of *cmu*s	# of features
41,674	10	1,258	246	13,171

Table 5.4: Information obtained from training of model **Direct-2**

5.2 Direct-2 Model Training

5.2.1 Procedure and Results

Direct-2 was trained using the data set *Base-0* in the following two steps:

1. Using *EM* iterations in GIZA++ to obtain *Viterbi* alignment of each pair of names in the training set of *Base-0*. The bootstrapping settings were the same as IBM SMT model training in [33]: 5 *EM* iterations of Model-1 followed by 5 of Model-2, 10 of Model-HMM and 10 of Model-4. Note that the direction of estimation is from *E* to *C* directly instead of the opposite direction in source-channel training;

2. Aligned training instances were then passed to *GIS* algorithm [4, 30] for training the MaxEnt model parameters. This fulfilled training the models considering the contextual features that can transliterate phoneme sequences of given English names into pinyin sequences.

After training, we obtained the outcome in Table 5.4 concerning the number of all the contexts, *cmu*s and features identified from the training corpus. Cut-off is the manually set threshold for ignoring features that occur very few times in the training data since their statistics may not be reliable [30]. The cut-off threshold of 10 means only those features that appear 10 times or more were considered to used for training the MaxEnt models. In addition, the possible *cmu*s corresponding to each English phoneme were randomly selected from the output of the *EM* training process being shown Table 5.5.

5.2.2 Discussions

Analysis of Table 5.5 revealed that there were two critical problems in **Direct-2** that can be further improved:

Deficiency-1

246 *cmu*s were identified (see Table 5.4), in which many were illegal clusters, such as /aw/, /axue/, /aie/, /af/,...,etc.. Actually, some were unfavorable “final-initial” clusters, which might or might not constitute legitimate pinyin

English Phoneme	Pinyin *cmu*s
AA	iang ch aw ao an uan ai aa eng ong ie ve ia w u o uo e ue ...
AE	ao an ai ab ie ia o uo ui a ε ian ei ang ...
AH	iang axue ou ao an uan aitian ai eng ata aotian ong iu ie ...
AO	ou aw ao an ai aie ong ie ia w u o uo i f e ue a ua iao ε ...
AW	uow ou uok uoh uof of aw ao hao an af ab azhsheng ash iu w u ...
AY	iaai jing uoy wei ay ar an hai ai uai ah uaiy aiy aiai aii aih rong ...
B	ch an uan in w u p f c b ε ...
CH	henshitin ch zh ie ia z x w t s q k j i h f d c iao ε ...
D	ch zh z t n j d c b
DH	ch z x t s q d c ε sh ...
EH	ou ao an ai eng ie ia o uo un i ui e ue ...
...	...

Table 5.5: Phoneme-*cum* mapping relationships discovered by *EM* training using GIZA++

sequences depending on the pinyin symbols followed. There is no means to prevent ill-formed *cmu*s from happening using GIZA++ training because it is completely data-driven. The *EM* algorithm is unbiased by treating each phonetic unit and all possible alignments equally. Unfavorable symbol-mappings and alignments are unavoidable if such mappings dominate the training data, e.g. the first phoneme /F/ maps to /f/ and the second one /R/ maps to /ul/. This results in illegal pinyin sequences and gives rise to a large number of uncertain *cmu*s. They turned out adding more uncertainties to phoneme mapping in testing.

Deficiency-2

Due to compound pinyin finals, two consecutive English phonemes may map to a single pinyin symbol, such as mapping from /AE N/ to /ang/ (see Figure 3.1). This is not allowed in both **Direct-1** (see Figure 4.1) and **Direct-2**. In these two models, one of the consecutive English phonemes must be mapped to ε. In the model **Direct-1**, they are considered mapping to ε in turn within the chunk and contribute to one more phoneme-level alignment (see Section 4.3). In GIZA++ training of **Direct-2**, they are considered as zero-fertility symbols, i.e. "words" with fertility zero in machine translation [1]. Note that these zero-fertility symbols become English phonemes as the direction of GIZA++ training is reversed in **Direct-2**. This approach is inaccurate because there are possible many-to-one mappings from English to Chinese phonemes.

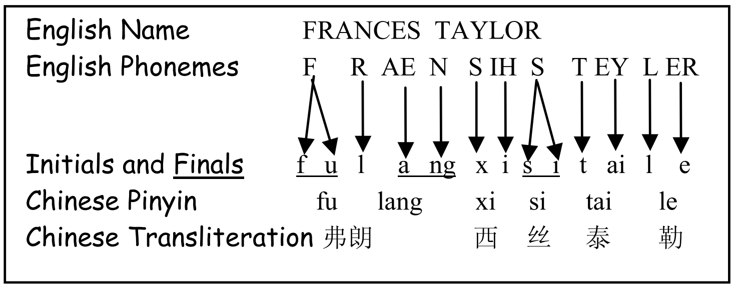

Figure 5.1: Our refined phoneme alignment scheme in direct transliteration modeling

5.3 Refining the Model Direct-2

These deficiencies motivate us to refine the model **Direct-2** by improving the alignment scheme and reducing the size of pinyin inventory.

5.3.1 Refinement Solutions

Based on the discussions concerning the deficiencies of **Direct-2**, we propose two related solutions to refine the model. This refined model is referred to as **Direct-2R**.

Solution-1

We replace the *EM* training of GIZA++ by the *EM* training initiated by the alignment of phoneme chunks for the model **Direct-1** (see Section 4.3). This aims to reduce the number of ill-formed *cmu*s by avoiding mappings across phoneme chunks. The alignment scheme based on phoneme chunks can also decrease the number of possible *cmu*s of each English phoneme, i.e. less and more deterministic class labels for each phoneme.

Solution-2

The linguistic knowledge about compound finals need not be ad-hoc in **Direct-2**. We can refine the data by decomposing the set of compound finals into multiple basic finals, e.g. from /ang/ into /a/ and /ng/, to reduce the size of vocabulary in the target language. The original mapping in Figure 3.1 is then broken into /AE/-to-/a/ and /N/-to-/ng/ as shown in Figure 5.1.

We carefully examined the 35 pinyin finals and identified 12 compound finals, which are listed in Table 5.6. We then decomposed compound finals

Compound Finals (12)	ang	eng	iao	ian	iang	ing
	iong	uai	uan	uang	ong	üan

Table 5.6: Compound finals identified by hand

Basic Finals (24)	a	o	e	ai	ei	ao	ou	er
	an	en	ng	i	ia	ie	iu	in
	u	ua	uo	ui	un	ü	üe	ün

Table 5.7: 24 basic final symbols in pinyin after refinement

into smaller units and ended up with a reduced set of final inventory with 24 basic units presented as Table 5.7.

5.3.2 Direct-2R Model Training

The training of the **Direct-2R** model was conducted using the similar procedure as that of **Direct-2** except that a refined phoneme-pinyin alignment scheme based on phoneme chunks and the *EM* training in Section 4.3.1 were applied to the step 1. In the refined alignment scheme a decomposition process that decomposed compound finals (see Table 5.6) to basic finals (see Table 5.7) was used in training.

We compared the **Direct-2R** training outcome with the **Direct-2** model training in terms of the number of contexts, *cmu*s and features identified. The results are shown in Table 5.8. The **Direct-2R** model identified more contexts and features, but generated fewer possible *cmu*s. The possible *cmu*s corresponding to each English phoneme were randomly selected from the output of *EM* training process and shown in Table 5.9. We note that the quality of the *cmu*s identified by **Direct-2R** model training have been improved as all the *cmu*s were either individual initial/finals or legal initial-final clusters. This justifies the effectiveness of the **Direct-2R** refinement over the **Direct-2**.

Model	Training Size	cut-off	# of contexts	# of *cmu*s	# of features
Direct-2	41,674	10	1,258	246	13,171
Direct-2R	41,674	10	1,282	195	13,933

Table 5.8: Information obtained from training of model **Direct-2** and **Direct-2R**

English Phoneme	Pinyin *cmu*s
AA	du ou luo yue ao ai ng mi lo le la ya xi sha wo we wa chi ie ...
AE	ao ai ng la ya wa ie ia u o uo lao i ui e a i ao ε ei ...
AH	sai da kai pi lie ou luo wei yue qia ao ai ni hai ng na ...
AO	xiao ou luo yue ao ai yu lo la wo we wa ie ia u o uo huo lao i ...
AW	ou luo bi ao ai wu you iu w u hu o uo i e a i ao ε ei ...
AY	da ci pa wei yue ba ao ai li yi ye ji iu ie ia v u ho o uo l i ...
B	ch bu bo wei bi ba bei ng bai y w s p n l g f e b a fu ε er ...
CH	du di de ci ch bo nu ng na zh ze yi le ye ke xia ji chu z y hu ...
D	ch xi tai z x t s q h d c te si sh
DH	lie ou wei bi ao ai hai nie lu nuo li yi lei le ye la ya iu ie ia ...
EH	ou wei ao ai lei le ye la ya xi wu qiao wo wa iu ie ve ia y u o ...
...	...

Table 5.9: Phoneme-*cum* mapping relationships discovered by *EM* training based on alignment of phoneme chunks

5.4 Evaluation

5.4.1 Search Algorithm

Given an English phoneme sequence $\{e_1, e_2, \ldots, e_n\}$, the conditional probability of generating the Chinese phonetic sequence $\{cmu_1, cmu_2, \ldots, cmu_n\}$ is given by:

$$p(cmu_1, cmu_2, \ldots, cmu_n | e_1, e_2, \ldots, e_n) \simeq \prod_{i=1}^{n} p(cmu_i | h_i) \qquad (5.10)$$

where $\{h_1, h_2, \ldots, h_n\}$ is the predefined context (history) with respect to each English phoneme. The transliteration probability $p(cmu|h)$ regarding the contextual history h can be estimated by Eq.(5.6), in which $p(cmu, h)$ obtains from Eq.(5.8). Eq.(5.8) is computed by using the feature functions and their respective model parameters obtained from the *GIS* training.

We applied "beam search" to this testing process [30]. Beam search is essentially a breadth-first algorithm, but can avoid the combinatorial explosion problem of breath-first search by expanding only a few most promising candidates at each level using certain heuristic. For each e_i in a testing English phoneme sequence $E = \{e_1, e_2, \ldots, e_n\}$, the algorithm maintains the N highest probability transliteration candidates up to and including e_i it sees in the sequence, where N is known as the "beam size". The search algorithm is shown in Algorithm 2. The beam size $N = 5$ was determined empirically [30] and the top-1 transliteration was finally used.

Algorithm 2 Beam search algorithm finding for best transliteration

```
1: Input: $e_1, e_2, \ldots, e_n$
2: Output: $cmu_1, cmu_2, \ldots, cmu_n$
3: Candidate Node: $b_{ij}$, the $j$th most probable transliteration candidates
   up to $e_i$, $1 \leq i \leq n, 1 \leq j \leq N$.
4: begin
5: Generate possible $cmu_1$ for $e_1$ by Eq.(5.6).
6: Find top-$N$ candidates $t_1^N$.
7: for $j = 1$ to $N$ do
8:     set $b_{1j} = t_j$
9: end for
10: for $i = 2$ to $n$ do
11:     for $j = 1$ to $N$ do
12:         Generate possible $cmu_i$ for $e_i$ by Eq.(5.6) given the transliteration
            context $b_{(i-1)j}$
13:         Append $cmu_i$ to $b_{(i-1)j}$ to generate new sequence $b_{ij}$
14:         Add $b_{ij}$ to candidate list
15:     end for
16:     Find top-$N$ candidates $t_1^N$ from candidate list
17:     for $j = 1$ to $N$ do
18:         set $b_{ij} = t_j$
19:     end for
20: end for
21: Return the most probable candidate $b_{n1}$
22: end
```

Systems		Baseline	Direct-1	Direct-2	Direct-2R
C.A.	Close	66.35%	63.17%	68.18%	76.97%
	Open	65.15%	62.61%	67.18%	75.08%
W.A.	Close	20.73%	13.16%	23.47%	36.19%
	Open	18.27%	11.34%	21.49%	32.50%

Table 5.10: Transliteration accuracies of the baseline, **Direct-1**, **Direct-2** and **Direct-2R**

5.4.2 Direct Transliteration Models vs. Baseline

Accuracy

We compared the performance of **Direct-2**, **Direct-2R**, **Direct-1** and the source-channel baseline with *C.A.* and *W.A.* measurements (see Section 3.3.2). Data set *Base-0* was used for the comparisons.

The results are shown in Table 5.10. **Direct-2** outperforms the source-channel baseline by about 2% in *C.A.* and about 3% in *W.A.*. It also outperforms **Direct-1** by about 6% in *C.A.* and about 12% in *W.A.*. This justifies our expectation on improving transliteration accuracies by considering contextual dependencies. The model **Direct-2R** demonstrates significant improvement over all the other models in all tests. Recall that in the model **Direct-2R**, we decomposed longer compound finals in pinyin into smaller sound units, i.e. basic finals, and aligned chunks of English phonemes with the corresponding chunks of pinyin symbols, prohibiting alignments across chunk borders. This could produce:

1. more precise mappings between English phonemes and mapping units in pinyin (*cmu*s);
2. less possible *cmu*s for each English phoneme, reducing uncertainties; and
3. less *cum*s with illegal pinyin syllables, leading to more legitimate pinyin sequences.

This justifies the effectiveness of (a). the alignment scheme based on phoneme chunks and (b). reduced "granularity" of Chinese phonemes, which helps for the precise alignment.

C.A. Distribution

The *C.A.* Distribution, i.e. the percentage of the number of transliterations distributing over different *C.A.* ranges (see Section 4.6.1), was measured as

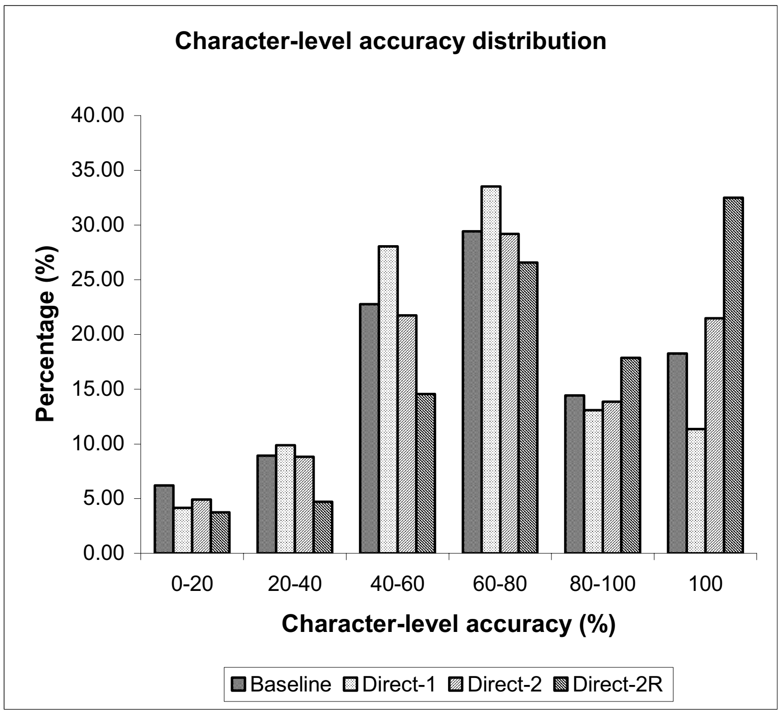

Figure 5.2: Comparison of baseline, **Direct-1**, **Direct-2** and **Direct-2R** on *C.A.* Distribution.

well. Figure 5.2 shows the results. For *C.A.* ranges of 0% to 20%, the baseline produced more transliterations than the other models; For *C.A.* ranging from 20% to 80%, **Direct-1** produced more transliterations throughout the three ranges than the others. In the remaining *C.A.* from 80% to 100%, **Direct-2R** produced more high-quality transliterations (see *C.A.* $\geq$ 80%) and considerably more correct transliterations (see *C.A.* = 100%).

Accuracy vs. Name Length

We also investigated how the different models differ from average length of correctly transliterated names. The length of a given name is represented by the number of phonemes it contains. For each model, we calculated the average length of correctly transliterated names (whose *C.A.* = 100%) in close and open tests. The averaged length of testing names were also calculated. The results are listed in Table 5.11.

We notice that the source-channel baseline somehow discriminates longer names, evidenced as the average length of correctly transliterated names is obviously shorter than that of all tested names, whereas other models are

	Close	Open
Baseline	5.02	4.91
Direct-1	5.28	5.22
Direct-2	5.27	5.27
Direct-2R	5.30	5.34
Avg. Length	5.36	5.40

Table 5.11: The averaged length of testing names vs. the averaged length of correctly transliterated names by different models

basically unbiased for name length as the average length of correctly transliterated names approaches to that of all averaged. This implies a good quality of our proposed models: we tend to be able to transliterate names of any length, not only being suitable for shorter ones. To justify our anticipation, we further conducted an experiment to reveal the relationship between *W.A.* and name length on different models using the results of open tests. The comparisons are presented as Figure 5.3.

We observe that the number of testing names distributed normally according to length. Most names have 5 or 6 phonemes. Basically, all the models tend to make more mistakes on longer names than shorter ones. However, it is evident that the baseline performs well on names with the length of 3, and turns out being worse sharply for names longer than 3 phonemes. Other models basically can persist their performance with the increase of name length from 4 to 7. This indicates that our direct approaches have apparent advantages on transliterating names of various lengths.

The reason is not very conclusive. We can give the intuitions roughly as follows: **Direct-2** and **Direct-2R** can capture longer distance dependencies of phonemes. Thus, their performances are less sensitive to length. However, this is not the case for **Direct-1**, which does not incorporate contextual features either, but practically performs better on length sensitivity than the baseline. This may result from the stochastic prediction for the positions of unaligned English phonemes and pinyin symbols in the baseline. Shorter names tend to have shorter transliterations, and thus tend to have fewer phonemes that should have been identified as zero-fertility and NULL-generated than longer ones. For shorter names, IBM SMT model therefore carries out fewer inaccurate operations, such as **AddZfert** and **AddNull** than for longer ones. This can explain why the baseline is good at shorter names. Because other direct models can deal with one-to-many phonetic mappings, they don't involve such a stochastic prediction mechanism. Consequently, the influence of name length is lighter.

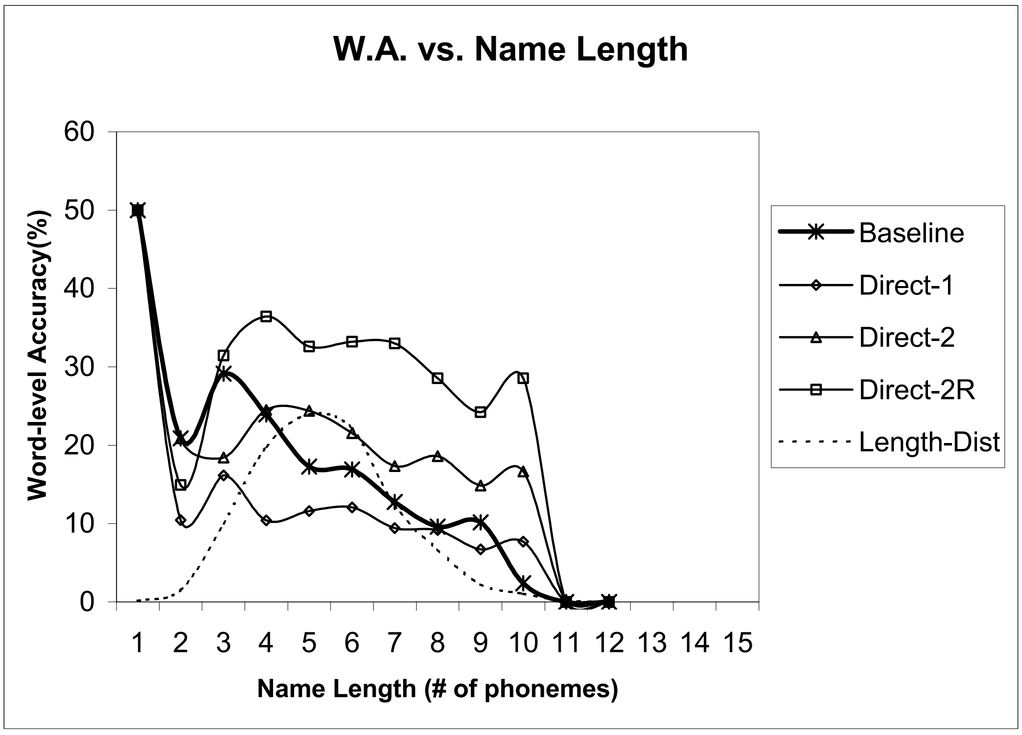

Figure 5.3: *W.A.* vs. name length under different models. The length of testing names is normal distributed.

5.4.3 Direct-2 vs. Direct-2R

We compared the performance of **Direct-2** and **Direct-2R** using *C.A.* and *W.A.* (see Section 3.3.2 for their learning curves, which is defined as accuracy varying with different data sizes. The experiment also enabled us to determine the appropriate training data size.

Data Preparation and Test Procedure

Experiments were carried out in multiple trials. Instead of directly using the data set *Base-0*, we extracted different instances in each trial of the experiments from the data pool which contains 46,305 name pairs. Data preparation proceeded as follows: In each trial, individual translation name pairs, i.e. instances, were randomly selected from the data pool to build 10 subsets. Each subset accounted for 10% to 100% (step=10%) of the total instances in the entire pool. In each subset, we used 90% of the instances for training and the remaining 10% for open test. Also the same number of instances (10%) were randomly selected from the training instances for close test. To investigate the influence of data sparseness, the procedures of training and testing were applied to the 10 subsets with different data sizes. And the performance was measured by the averaged accuracy (*C.A.* and *W.A.*) of 50 trials of the experiments. *C.A.* Distribution was tested by the averaged values of the 50 trials on 100% data size only. These data preparation and testing procedures were designed to smoothen certain bias that often existed in individual data sets.

Table 5.12 shows the highest and lowest accuracies of **Direct-2** and **Direct-2R** in different trials together with the corresponding data sizes. *C.A.* and *W.A.* were averaged over the number of testing names. We noted that the maximum/mininum of *C.A.* and *W.A.* may happen regardless of the data size. It implies that the quality of data in certain subsets can affect the outcome of each trial. This is the reason why we have to average the accuracies on multiple trials to smoothen the discrepancies caused by different data quality in individual subsets.

Learning Curve: Accuracy vs. Data Size

Figure 5.4 shows the average *C.A.* and *W.A.* of **Direct-2** and **Direct-2R** over different data sizes. **Direct-2R** significantly outperforms **Direct-2** on all tests for different subsets.

We did not see serious data sparseness problem. This is likely because our data pool was large enough, with 46,305 name pairs. However, light data sparseness was observed when less than 40% of the data pool (around

Accuracy		Direct-2		Direct-2R	
		Close	Open	Close	Open
C.A.	Max(%)	72.48	69.26	**79.23**	76.83
	Size	10%	10%	**20%**	90%
	Min(%)	66.51	**63.23**	72.97	70.93
	Size	60%	**10%**	10%	10%
W.A.	Max(%)	28.08	24.48	**40.24**	36.11
	Size	10%	50%	**20%**	90%
	Min(%)	21.17	**14.47**	30.98	24.41
	Size	10%	**10%**	10%	10%

Table 5.12: Marginal accuracies achieved by **Direct-2** and **Direct-2R**

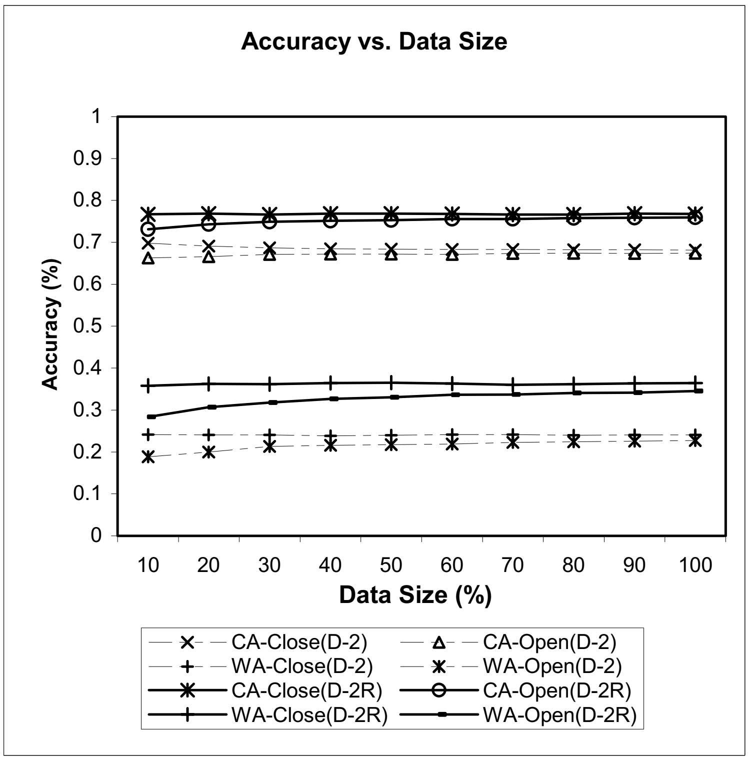

Figure 5.4: Comparisons of **Direct-2** and **Direct-2R** on *C.A.* and *W.A.* over different data sizes. **D-2** and **D-2R** refer to model **Direct-2** and **Direct-2R** respectively

18,522 instances) was used for **Direct-2R** training (90% of 18,522 training instances) and testing (10% of 185,22 test instances). In **Direct-2**, we observed similar data sparseness when 30% of the entire data set (around 13,891 instances) was used. These data sizes, i.e. about 13,891 instances for **Direct-2** and 18,522 instances for **Direct-2R**, should be enough for training the models. However, this indicates that **Direct-2R** is more sensitive to sparse data than **Direct-2**. The reason is that fewer *cmu*s of each English phoneme are discovered in **Direct-2R** than in **Direct-2**. Intuitively, when small training set were used, not enough *cmu*s could be discovered by the *EM* training based on alignment of phoneme chunks. Although the *cmu*s discovered were finer than those of GIZA++ training, they could not make up of "full-scale" class labels for English phonemes, which caused the *MaxEnt* models to wrongly predict yet "unseen" *cmu*s by making use of this fewer number of "seen" *cmu*s during testing. This can explain why **Direct-2R** suffers from data sparseness problem more seriously.

5.4.4 Experiments on Direct-2R

Transliteration Quality

We analyzed **Direct-2R** to study the quality of transliterations. We randomly chose 100 sample testing names with $C.A. \leq 20\%$ to qualitatively examine their English pronunciations, machine-generated transliterations and the standard Chinese translations. These names are partially listed in Table 5.13. We recognized 68 foreign names that should not be phonetically transliterated but should be translated based on meaning, such as Japanese, Korean or other Southeast Asians' names. Most of the remaining ones are "irregularly" transliterated names, such as "Rieth", "Hayer", "Haim", "Flex", etc. Since we have no idea concerning their original language, it is difficult to judge their human-generated standard transliterations. But the machine-generated transliterations are evidently closer to their English pronunciations than the standard ones. Note that non-English names possibly, but not always, can be transliterated in terms of their original pronunciations instead of English pronunciations. For instance, the transliteration of "John" (/JH AA N/), " 约翰 " (/yue han/), is produced from its Hebrew pronunciation directly. Hence, it would be better to first classify the training instances according to their language origins. This, however, is beyond the scope of this thesis.

We also studied the transliterations of 100 randomly chosen names with $C.A.$ of more than 80% but less than 100%. Some sample names are listed in Table 5.14. Qualitatively, they all present tiny distance from the standard

Original Name	Pronunciation	Machine Trans.	Standard Trans.	C.A.(%)
Voth	V AA TH	wa si	fu te (福特)	0.000
Honcho	HH AO N CH OW	heng huo	ben die (本蝶)	16.67
Pace	P EY S	pei qie	pa cai (帕采)	20.00
Rieth	R AY AH TH	li ao si	li te (里特)	0.000
Fujitsu	F UW JH IH T S UW	fu ji ce	ge jin (葛津)	20.00
Sag	S AE G	sa ge	sa (萨)	0.000
Tokunaga	T OW K UW N AA G AH	tuo ke na jia	de chang (德长)	0.000
Yoho	Y OW HH OW	yue huo	rong feng (蓉峰)	0.000
Hiromasa	HH IH R OW M AA S AH	xi luo ma sa	bo ya (博雅)	0.000
Gyosai	G Y OW S EY	ge luo sa	yu cai (鱼菜)	0.000
Bag	B AE G	ba ge	ba (巴)	0.000
Thyme	TH AY M	sai mu	di mei (蒂梅)	20.00
Upshur	AH P SH ER	a pu xiao	e pu she (厄普舍)	16.67
Haim	HH AY M	hai mu	an (安)	0.000
Pet	P EH T	P EH T	bei (贝)	0.000
Shaefer	SH EY F ER	sha fe	xie fu (谢弗)	20.00
Hayer	HH EY ER	hai er	a ye (阿耶)	0.000
Motyka	M AA T AY K AH	ma tai xi	mo di ka (莫蒂卡)	16.67
Flex	F L EH K S	fu lai ke si	fu lai (弗莱)	20.00
Yap	Y AE P	ya pei	ru (入)	0.000
...	...	...	...	...

Table 5.13: Randomly selected sample transliterations with $C.A. \leq 20\%$

transliterations. Many of them are even phonetically closer to the corresponding English pronunciations than the standard ones. Without profound linguistic knowledge, one could not distinguish the subtle qualitative differences between transliterations produced by machine and human. However, they are not identical to the defacto standards after all. Thus, in human subject tests, transliterations with similar quality like this could be considered "phonetically equivalent but misspelled" [19]. A precise objective judgment method for the transliteration quality is still left unexplored.

Phoneme Accuracy

We adopted an approximated approach to examine the phoneme conversion accuracy and top confusions among mapping units in pinyin: Assuming the *Viterbi* alignments obtained during training were correct, the mapping of individual phonemes would also be correct. We used all the 46,305 instances in the data pool for training and testing. The $C.A.$ and $W.A.$ achieved were 76.79% and 36.43% respectively. We then compared the test outputs, i.e. transliterations aligning to their English origins, and the alignments produced by *EM* algorithm during the training process. Conversion accuracies of English phonemes are listed in Table 5.15.

We found that 2/3 of 24 consonants and 1/4 of 16 vowels are among top-

Original Name	Pronunciation	Machine Trans.	Standard Trans.	C.A.(%)
Cotroneo	K OW T R OW N IY OW	ke te luo ni ao	ke te luo nei ao (科特罗内奥)	91.67
Schmaus	SH M AW Z	shi ma si	shi mao si (施毛斯)	87.50
Krzeminski	K R AH M IH N S K IY	ke la ming si ji	ke re ming si ji (克热明斯基)	83.33
Bergfeld	B ER G F EH L D	bo ge fei er de	bei ge fei er de (贝格菲尔德)	83.33
Priscilla	P R AH S IH L AH	pu lu xi la	pu li xi la (普丽西拉)	87.50
Silverthorn	S IH L V ER TH AO R N	xi er wo suo en	xi er fu suo en (西尔弗索恩)	81.82
Klages	K L EY JH AH Z	ke la re si	ke la ge si (克拉格斯)	87.50
Cullins	K AH L IH N Z	ke lin si	ka lin si (卡林斯)	85.71
Pellett	P EH L AH T	pei la te	pei li te (佩利特)	85.71
Hayworth	HH EY W ER TH	ai wo si	hai wo si (海沃思)	85.71
Putting	P AH T IH NG	pa ting	pi ting (皮廷)	83.33
Drees	D R IY Z	de li si	de lei si (德雷斯)	85.71
Garnett	G AA R N EH T	jia nei te	jia ni te (加尼特)	85.71
Cronquist	K R AA N K W IH S T	ke lan kui si te	ke long kui si te (克龙奎斯特)	84.62
Cusumano	K UW S UW M AA N OW	ku zu ma nuo	ku su ma nuo (库苏马诺)	88.89
Perkovic	P ER K AH V IH CH	pei ke wei qi	pei er ke wei qi (佩尔科维奇)	83.33
Lagardere	L AA G AA R D IH R	la jia di er	la jia dai er	90.00
Valley	V AE L IY	wa li	wa lai (瓦莱)	80.00
Delmonico	D EH L M AA N IY K OW	de er ma ni ke	de er mo ni ke (德尔莫尼科)	90.00
Steward	S T UW ER D	shi tu er de	si tu er de (斯图尔德)	87.50
...	...	...	...	...

Table 5.14: Randomly selected sample transliterations with $80\% \leq C.A. < 100\%$

20 of phoneme accuracies. This indicates that transliterations of consonants are more accurate than vowels. We obtained average accuracies of consonants and vowels with 76.36% and 67.78% respectively from Table 5.15. The reason is straightforward: consonants are all monophonic while vowels vary from monophthongs, diphthongs to even triphthongs. Vowels complicate the mapping relationships with pinyin symbols and may lead to more possible *cmu*s than consonants do. A possible solution for future work can be to somehow decrease the "granularity" of phonetic representations on both languages to yield finer mapping correspondences between phonemes or phoneme chunks.

Top-20 *cmu*s in pinyin that are most frequently confused with others are shown in Table 5.16. Their corresponding English phonemes and confusion frequencies are also shown. The confusions are partly resulted from the fact that people did not abide by consistent regulations, especially on sounds with high transliteration ambiguities. For example, pinyin sound /p/ is often confused by /b/ for the given English phoneme /P/. There were 579 transliteration mistakes on phoneme /P/, where 347 of them wrongly converted /P/ to /p/ instead of to /b/, which ought to be the correct target, and the confusion rate is 58.12%.

Phoneme	Top-20 Accuracy	Phoneme	Lower-19 Accuracy
F	97.73%	UH	73.36%
B	96.02%	HH	72.04%
L	94.59%	EH	70.92%
D	92.94%	AA	68.62%
M	92.62%	Z	64.78%
AE	89.95%	UW	64.66%
R	88.91%	AO	64.31%
P	86.96%	DH	63.64%
K	86.81%	ER	63.12%
IH	84.89%	OY	62.24%
T	83.55%	CH	60.18%
N	81.82%	AW	60.06%
W	81.51%	TH	58.86%
V	81.04%	AH	57.76%
OW	80.76%	AY	57.05%
G	80.48%	Y	56.71%
S	80.46%	JH	43.36%
IY	78.26%	EY	40.77%
NG	77.95%	ZH	35.35%
SH	74.21%		

Table 5.15: Phonemes accuracy rankings

Pinyin Confusion	English Phoneme	Confusion Frequency	Confusion Rate
b → p	P	347/597	58.12%
ng → n	N	1719/3398	50.59%
t → s	DH	24/56	42.86%
ng → n	NG	180/460	39.13%
ε → er	R	589/1622	36.31%
b → w	V	197/563	34.99%
x → sh	SH	179/517	34.62%
i → ai	AY	390/1170	33.33%
shi → si	S	795/2473	32.15%
ε → de	D	191/599	31.88%
te → si	TH	96/353	27.20%
j → g	G	266/1050	25.33%
u → uo	OY	36/148	24.32%
d → t	T	431/1892	22.78%
ε → er	L	203/902	22.51%
i → u	UH	42/191	21.99%
ei → i	IY	551/2510	21.95%
a → h	HH	221/1022	21.62%
fei → fu	F	19/89	21.35%
w → b	B	65/313	20.77%

Table 5.16: Pinyin mapping units with top confusions and their corresponding English phonemes

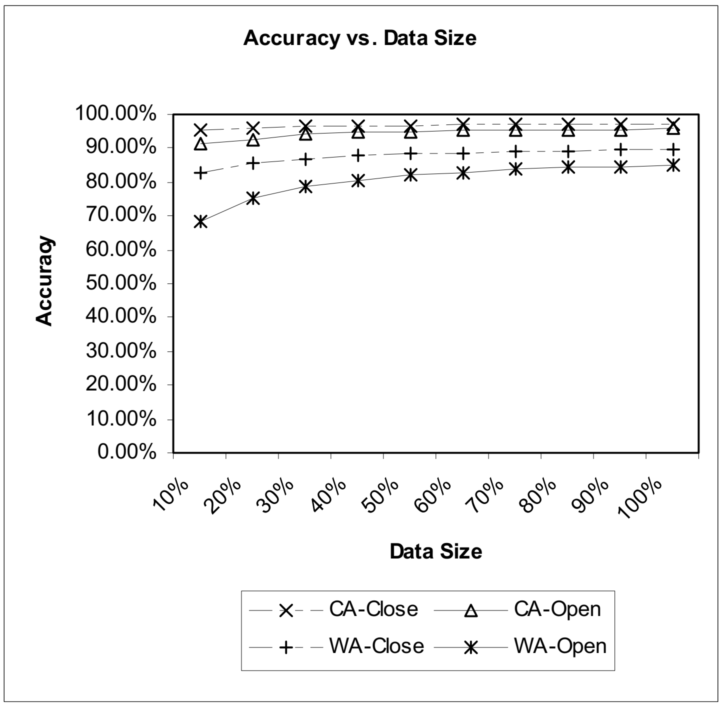

Figure 5.5: The learning curve of **Direct-2R** on trimmed data set *Base-1*

Performance on Trimmed Data and Remarks

Before we conclude this chapter, we demonstrate the system's performance of model **Direct-2R** with "friendly" data. The data set we used contains a large number noisy data like irregularly transliterated names from a variety of languages. It's been testified that these "unfriendly" names contributed significant errors to transliterations [33], evidenced as the decrease of errors after the names with low alignment scores were wiped off from the training set. We propose to learn how well our direct approach can achieve with regular names, such as how its learning curve is like when not disturbed by noises.

We refined the data pool by eliminating all the incorrectly transliterated names. This was done by training and testing **Direct-2R** on all the 46,305 instances. We obtained 16,948 very regular instances, which is called *Base-1* hereafter. We then partitioned *Base-1* into 10 subsets according to the method we processed *Base-0* in Section 5.4.3 during each of the 50 trials for training and testing. The resulting *C.A.* and *W.A.* were averaged over all the trials. Figure 5.5 presents the learning curve of **Direct-2R** on *Base-1*.

The model achieves remarkably higher accuracies on the trimmed data set. On 100% data size, the *C.A.* reaches 97.08% and 95.90% in close test and open test respectively, and the *W.A.* are 89.53% and 85.32%. However, we should emphasize several remarks as follows:

1. From these tests, we can notice evident data sparseness problem as the data size is smaller than 50% of 16,948 instances. Usually, it would be hard to have about 8,500 name pairs for training the model. Thus, techniques addressing data sparseness should be incorporated into the model.

2. Under this ideal data set, we can see the maximal potential of the model. There are still rooms to improve the model possibly by introducing additional dependencies, such as longer contextual histories or the language model. This is up to the further study concerning the influence of different features on the transliteration effectiveness.

3. It is necessary to improve the model's performance under the richness of noisy data.

5.5 Chapter Summary

We have addressed several critical problems suffered by the source-channel based English-to-Chinese transliteration model. We then proposed the improved direct transliteration model **Direct-2**, which supported one-to-many alignment mappings. The model is simplified by using mapping units to reduce the number of model parameters. That finally will lead to one-to-one alignments.

The implementation of **Direct-2** did not make use of information embedded in the target language model. This was balanced by using the contextual information of a specific English phoneme and the corresponding aligned Chinese pinyin symbols. Maximum entropy modeling was employed for this purpose and *GIS* training algorithm was used. Experiments showed its superiority over the source-channel baseline and **Direct-1**.

Further refinement by **Direct-2R** was achieved by precise alignment of phoneme chunks and by decomposition of larger phonetic units, i.e. compound finals of pinyin, into basic finals with smaller "granularity". The experimental results strongly supported our expectations.

□ **End of chapter.**

Chapter 6

Conclusions

This chapter summarizes the contributions of this thesis. It outlines possible applications of machine transliteration and gives an outlook for future research on this interesting topic.

6.1 Thesis Summary

We have proposed to use unsupervised machine learning techniques for modeling phoneme-based English-to-Chinese transliteration problem. Statistical transliteration modeling has a strong basis in information theory. It can remove many ad hoc procedures from traditional rule-based machine transliteration techniques. In addition, we have shown that the proposed direct transliteration approaches have obvious advantages over IBM SMT model, which is based on the state-of-the-art source-channel framework. We have evaluated and compared both the direct and source-channel based models and obtained strong evidences to support our hypotheses. There are three main contributions in our research:

- We have identified major deficiencies of the source-channel based model for English-to-Chinese transliteration by reproducing the implementation described in [33]. The source-channel based model, which uses reversed prior conditional probability, is unable to realize one-to-many symbol mappings between phonetic units from source to target language. Transliteration is modeled as a stochastic process with randomized parameters, such as zero-fertility and NULL-generated symbols, to represent unaligned phonetic units. The mechanism is error-prone as random reproduction of zero-fertility symbols in target sequence and insertion of NULL-generated symbols in source sequence could not effectively predict the frequent un-transliterated English phonemes and the mappings from single English phonemes to initial-final clusters. This restriction has been overcomed by the proposed direct transliteration model, i.e. the direction of prior probability estimation coincides with the transliteration direction.

- We proposed the **Direct-1** model to cope with one-to-many symbolic mapping with a direct prior and alignment scheme of phoneme chunks. Initial-final clusters were introduced as mapping units in the Chinese side. They could be dynamically identified and appended to the target phonetic vocabulary during training. The direct method was implemented using a weighted finite state transducer for the transliteration model plus a finite state acceptor based on the bigram language model of the target syllables. Compared with the source-channel baseline, the performance of **Direct-1** was slightly worse. Nonetheless, **Direct-1** is more simple and flexible for extension.

- We also proposed an enhanced direct transliteration model, namely **Direct-2**. Maximum entropy formalism was adopted to incorporate longer contextual dependencies among phonetic symbols. Although **Direct-2** excluded the language model, it still achieved higher accuracies than other approaches. We refined the **Direct-2** by using alignment based on phoneme chunks with finer pinyin mapping units. The refined model, i.e. **Direct-2R**, performed best. **Direct-2** did not only cater for one-to-many symbol mapping, but also approximated many-to-one mappings by reducing the granularity of target phonemes. This led to more precise symbolic alignment.

English-to-Chinese transliteration involves both one-to-many as well as many-to-one symbolic mapping between English and Chinese. Experiments have shown that the one-to-many alignment scheme plus contextual dependencies using direct models is better than the many-to-one by source-channel. It can be anticipated that the combination of the both would be more superior.

6.2 Cross Language Applications

The use of machine transliteration in cross language applications is imminent and promising. Currently, English-Chinese machine transliteration methods are almost all proposed for specific applications, such as information retrieval [8, 22, 23, 33], acquisition or extraction of equivalent word pairs from parallel corpus [14, 20], and construction of named entity translation dictionary [13, 34].

Not only that, automatic transliteration can be directly applied to machine-aided translation on names to alleviate human labors. It could also be applied to adaptation of empirically sound data to provide best suggestions to human translators on given names. In addition, although we were mainly concerned with Mandarin, dialect-specific knowledge is independent of our model. As such, our method will be applicable to other Chinese dialects, such English-Cantonese pairs. It could be trained using regional "cultural settings" to handle transliterations of specific Chinese dialects and help find transliteration equivalents of the same foreign name. In principle, our approach can also be applied to any language pairs that are composed of regular consonants and vowels, except for Semitic languages, e.g. Arabic, which lack "short vowels" in their written forms [2, 3].

6.3 Future Work and Directions

As far as we concern, machine transliteration for English-Chinese language pair is an undeveloped research area. There are still many unresolved problems and much work is left unexplored. As with our research, we outline the future work as follows:

1. Although direct approaches are advantageous over source-channel based methods on using one-to-many mapping from source to target and on accomodating contextual features, it may suffer from the hardship considering many-to-one mapping. For superior performance, it needs to approximate many-to-one mapping by means of decomposing the mapping to multiple one-to-one mappings, which was achieved by reducing the granularity of target symbols in **Direct-2R**. However, this may be ineffective for other target languages, in which there are not so many compound phonetic symbols as Chinese Romanization systems. Thus, a more general and effective model for many-to-many symbolic alignment should be developed.

2. Due to the decomposition of compound pinyin symbols, fewer and finer *cmu*s are identified in **Direct-2R** model. The model becomes more vulnerable to the paucity of data. Thus, effective smoothing techniques are required for the **Direct-2R** model to overcome data sparseness problem.

3. The use of different contextual dependencies has not been carefully studied and compared. It is unclear how well the features defined in the

improved direct models could suit for transliteration problems. Therefore, it would be interesting to explore the influence of different sets of features selected for **Direct-2** and **Direct-2R**.

4. There are many rooms to improve the **Direct-1** model by means of amendments on language modeling and search algorithm. There are constant arguments that the mathematical foundation of **Direct-1** results in its worse accuracy than the source-channel model supported by Bayes' theorem. However, it is not conclusive unless optimal solutions of the both models could be found and compared. Also, an experiment in machine translation showed that the form of **Direct-1** used for search did not affect the quality of translation results in source-channel based model [27].

5. No consent has been achieved in machine translation community whether the source-channel based IBM SMT model could obtain the optimality by the combination of translation model and language model. Perplexity is often adopted to measure the goodness of translation model in training and testing. However, it is unclear if perplexity could really reflect the effectiveness of translation [1]. Thus, it is an ongoing research to explore different combinations and iteration schedules of the sub-models (Model-1 to Model-5) in IBM SMT to find the optimal hypotheses. In direct transliteration modeling, MaxEnt theorem can guarantee that the learning algorithm always robustly converges to the maximum entropy probability distribution and provides better fit of the data. It would be useful to learn the optimal characteristics of the direct models and compare with that of the baseline model.

6. We can readily apply the proposed direct transliteration models to Chinese-to-English back-transliteration. Although we currently haven't conducted experiments to verify its effectiveness on the backward direction application, we believe that it is simply a matter of reversing the order of source and target without significant modification of the model. The source-channel model has intrinsic advantage for back-transliteration since its reversed priori probability estimation coincides with the original direction in producing transliterations by human translators (see remarks in Section 2.2.4). It would be interesting to make comparisons with the direct models for the effectiveness.

Finally, we would like to prospect the future direction on this research. One of the most exciting directions of machine transliteration is to propose a language independent model that can universally fulfill transliterating names given any language pairs. The premise of its realization depends not only on a model, but also on effective schemes on universal phonemic representation for all human languages. International Phonetic Alphabet (IPA) is widely used as such a representation for human speakers, but the universal computer-readable phonetic alphabet has just been developed in recent several years, which is known as *SAMPA* (Speech Assessment Methods Phonetic Alphabet). A SAMPA transcription is designed to be uniquely parsable. As with the ordinary IPA, a string of SAMPA symbols does not require space between successive symbols. Unlike other proposals for mapping the IPA to ASCII, SAMPA represents the outcome of collaboration and consultation among speech researchers in many different countries. The SAMPA transcription symbols have been developed by or in consultation with native speakers of every language to which they have been applied. Currently, it has been applied to 24 major human languages including Chinese, and the scope is continuing to enlarge. A number of research on speech technologies has been conducted based on this set of alphabets as internal phonetic representations. SAMPA will be a powerful tool for machine transliteration as well.

□ **End of chapter.**

Appendix A

IPA-ARPABET Symbol Mapping Table

IPA Symbol	ARPAbet Symbol	Word	IPA Transcription	ARPAbet Transcription
[i]	[iy]	lily	[ˈlɪli]	[l ih l iy]
[ɪ]	[ih]	lily	[ˈlɪli]	[l ih l iy]
[eɪ]	[ey]	daisy	[ˈdeɪzi]	[d ey z i]
[ɛ]	[eh]	poinsettia	[pɔɪnˈsɛɾiə]	[p oy n s eh dx iy ax]
[æ]	[ae]	aster	[ˈæstɚ]	[ae s t axr]
[ɑ]	[aa]	poppy	[ˈpapi]	[p aa p i]
[ɔ]	[ao]	orchid	[ˈɔrkɨd]	[ao r k ix d]
[ʊ]	[uh]	woodruff	[ˈwʊdrʌf]	[w uh d r ah f]
[oʊ]	[ow]	lotus	[ˈloʊɾəs]	[l ow dx ax s]
[u]	[uw]	tulip	[ˈtulɨp]	[t uw l ix p]
[ʌ]	[uh]	buttercup	[ˈbʌɾɚˌkʌp]	[b uh dx axr k uh p]
[ɝ]	[er]	bird	[ˈbɝd]	[b er d]
[aɪ]	[ay]	iris	[ˈaɪrɨs]	[ay r ix s]
[aʊ]	[aw]	sunflower	[ˈsʌnflaʊɚ]	[s ah n f l aw axr]
[oɪ]	[oy]	poinsettia	[pɔɪnˈsɛɾiə]	[p oy n s eh dx iy ax]
[ju]	[y uw]	feverfew	[fivɚfju]	[f iy v axr f y u]
[ə]	[ax]	woodruff	[ˈwʊdrəf]	[w uh d r ax f]
[ɨ]	[ix]	tulip	[ˈtulɨp]	[t uw l ix p]
[ɚ]	[axr]	heather	[ˈhɛðɚ]	[h eh dh axr]
[ʉ]	[ux]	dude[1]	[dʉd]	[d ux d]

Figure A.1: The correspondence between IPA symbols and ARPABET symbols (vowels)

IPA Symbol	ARPAbet Symbol	Word	IPA Transcription	ARPAbet Transcription
[p]	[p]	parsley	[ˈparsli]	[p aa r s l iy]
[t]	[t]	tarragon	[ˈtærəgɑn]	[t ae r ax g aa n]
[k]	[k]	catnip	[ˈkætnɨp]	[k ae t n ix p]
[b]	[b]	bay	[beɪ]	[b ey]
[d]	[d]	dill	[dɪl]	[d ih l]
[g]	[g]	garlic	[ˈgɑrlɨk]	[g aa r l ix k]
[m]	[m]	mint	[mɪnt]	[m ih n t]
[n]	[n]	nutmeg	[ˈnʌtmɛg]	[n ah t m eh g
[ŋ]	[ng]	ginseng	[ˈdʒɪnsɨŋ]	[jh ih n s ix ng]
[f]	[f]	fennel	[ˈfɛnl̩]	[f eh n el]
[v]	[v]	clove	[kloʊv]	[k l ow v]
[θ]	[th]	thistle	[ˈθɪsl̩]	[th ih s el]
[ð]	[dh]	heather	[ˈhɛðɚ]	[h eh dh axr]
[s]	[s]	sage	[seɪdʒ]	[s ey jh]
[z]	[z]	hazelnut	[ˈheɪzl̩nʌt]	[h ey z el n ah t]
[ʃ]	[sh]	squash	[skwɑʃ]	[s k w a sh]
[ʒ]	[zh]	ambrosia	[æmˈbroʊʒə]	[ae m b r ow zh ax]
[tʃ]	[ch]	chicory	[ˈtʃɪkɚi]	[ch ih k axr iy]
[dʒ]	[jh]	sage	[seɪdʒ]	[s ey jh]
[l]	[l]	licorice	[ˈlɪkɚɨʃ]	[l ih k axr ix sh]
[w]	[w]	kiwi	[ˈkiwi]	[k iy w iy]
[r]	[r]	parsley	[ˈpɑrsli]	[p aa r s l iy]
[j]	[y]	yew	[yu]	[y uw]
[h]	[h]	horseradish	[ˈhɔrsrædɪʃ]	[h ao r s r ae d ih sh]
[ʔ]	[q]	uh-oh	[ʔʌʔoʊ]	[q ah q ow]
[ɾ]	[dx]	butter	[ˈbʌɾɚ]	[b ah dx axr]
[ɾ̃]	[nx]	wintergreen	[wɪɾ̃ɚgrin]	[w ih nx axr g r i n]
[l̩]	[el]	thistle	[ˈθɪsl̩]	[th ih s el]

Figure A.2: The correspondence between IPA symbols and ARPABET symbols (consonants)

Bibliography

[1] Y. Al-Onaizan, J. Curin, K. K. M. Jahr, J. Lafferty, D. Melamed, F. J. Och, D. Purdy, N. A. Smith, and D. Yarowsky. Statistical machine translation. In *Final Report of Johns Hopkins University 1999 Summer Workshop on Language Engineering*, 1999.

[2] Y. Al-Onaizan and K. Knight. Machine transliteration of names in arabic text. In *Proc. of the ACL Workshop on Computational approaches to Semitic Languages*, 2002.

[3] M. Arbabi, S. M. Fischthal, V. C. Cheng, and E. Bart. Algorithms for arabic names transliteration. *IBM Journal of Research and Development*, 38(2), Mar. 1994.

[4] A. L. Berger, S. A. D. Pietra, and V. J. D. Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, Mar. 1996.

[5] S. Bilac and H. Tanaka. Improving back-transliteration by combining information sources. In *Proc. of the First Internatinal Joint Conference on Natural Language Processing*, pages 542–547, Mar. 2004.

[6] E. Brill. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 21(4):543–565, Dec. 1995.

[7] P. F. Brown, S. A. D. Pietra, V. J. D. Pietra, and R. L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, June 1993.

[8] H. H. Chen, S. J. Huang, Y. W. Ding, and S. C. Tsai. Proper name translation in cross-language information retrieval. In *Proc. of the 17th COLING and 36th Annual Meeting of ACL*, pages 232–236, 1998.

[9] P. Clarkson and R. Rosenfeld. Statistical language modeling using the cmu-cambridge toolkit. In *Proc. of the 5th European Conference*

on Speech Communication and Technology (EUROSPEECH 97), pages 2707–2710, 1997.

[10] J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annuals of Mathematical Statistics*, 43:1470–1480, 1972.

[11] D. Eppstein. Finding the k shortest paths. In *Proc. of the 35th IEEE Symposium on the Foundations of Computer Science*, pages 154–165, 1994.

[12] U. Germann, M. Jahr, K. Knight, D. Marcu, and K. Ya-mada. Fast decoding and optimal decoding for machine translation. In *Proc. of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 228–235, July 2001.

[13] F. Huang and S. Vogel. Improved named entity translation and bilingual named entity extraction. In *Proc. of the 4th IEEE International Conference on Multimodal Interfaces (ICMI02)*, pages 253–258, Oct. 2002.

[14] F. Huang, S. Vogel, and A. Waibel. Extracting named entity translingual equivalence with limited resources. *ACM Transactions on Asian Language Information Processing (TALIP)*, 2(2):124–129, June 2003.

[15] K. S. Jeong, S. H. Myaeng, J. S. Lee, and K. S. Choi. Automatic identification and back-transliteration of foreign names for information retrieval. *Information Processing and Management*, 35(4):523–540, 1999.

[16] H. L. Jin and K. F. Wong. A chinese dictionary construction algorithm for information retrieval. *ACM Transactions on Asian Language Information Processing (TALIP)*, 1(4):281–296, Dec. 2002.

[17] I. H. Kang and C. C. Kim. English-to-korean transliteration using multiple unbounded overlapping phoneme chunks. In *Proc. of the 17th Conference on Computational Linguistics*, pages 418–424, July 2000.

[18] K. Knight. A statistical mt tutorial workbook. Aug. 1999.

[19] K. Knight and J. Graehl. Machine transliteration. *Computational Linguistics*, 24(4):599–612, Dec. 1998.

[20] C. J. Lee and J. S. Chang. Acquisition of english-chinese transliterated word pairs from parallel-aligned texts using a statistical machine transliteration model. In *Proc. of the HLT/NAACL 2003 workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, pages 96–103, May 2003.

[21] J. S. Lee and K. S. Choi. English to korean statistical transliteration for information retrieval. *Computer Processing of Oriental Languages*, 12(1):17–37, 1998.

[22] W. H. Lin and H. H. Chen. Backward machine transliteration by learning phonetic similarity. In *Proc. of 6th Conference on Natural Language Learning (CoNLL)*, pages 139–145, Aug. 2002.

[23] H. M. Meng, W. K. Lo, B. Chen, and K. Tang. Generating phonetic cognates to handle named entities in english-chinese cross-language spoken document retrieval. In *Proc. of the Automatic Speech Communication Recognition and Understanding Workshop*, Dec. 2001.

[24] M. Mohri and M. Riley. Weighted finite-state transducers in speech recognition (tutorial) [part i, part ii]. In *International Conference on Spoken Language Processing 2002 (ICSLP 02)*, Sept. 2002.

[25] F. J. Och and H. Ney. A comparison of alignment models for statistical machine translation. In *Proc. of the 18th Conference on Computational Linguistics*, pages 1086–1090, July 2000.

[26] F. J. Och and H. Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proc. of the 40th Annual Meeting of the Association of Computational Linguistics*, pages 295–302, July 2002.

[27] F. J. Och, C. Tillmann, and H. Ney. Improved alignment models for statistical machine translation. In *Proc. of the Joint Conf. of Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 20—28, June 1999.

[28] J. H. Oh and K. S. Choi. An english-korean transliteration model using pronunciation and contextual rules. In *Proc. of the 19th International Conference on Computational Linguistics (COLING)*, 2002.

[29] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of IEEE*, (2), Feb. 1989.

[30] A. Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *Proc. of the first Empirical Methods in Natural Language Processing Conference(EMNLP)*, pages 133–142, May 1996.

[31] H. N. S. Vogel and C. Tillmann. Hmm-based word alignment in statistical translation. In *Proc. of the 16th International Conference on Computational Linguistics*, pages 836–841, Aug. 1996.

[32] B. G. Stalls and K. Knight. Translating names and technical terms in arabic text. In *Proc. of COLING/ACL Workshop on Computational Approaches to Semitic Languages*, 1998.

[33] P. Virga and S. Khudanpur. Transliteration of proper names in cross-lingual information retrieval. In *Proc. of the ACL Workshop on Multilingual and Mixed-languge Named Entity Recognition*, pages 57–64, July 2003.

[34] S. Wan and C. M. Verspoor. Automatic english-chinese name transliteration for development of multilingual resources. In *Joint Meeting of 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics*, pages 1352–1356, 1998.