UPPSALA
UNIVERSITET

Machine Transliteration of Names from Different Language Origins into Chinese

Yan Shao

Uppsala University
Department of Linguistics and Philology
Master's Programme in Language Technology
Master's Thesis in Language Technology

September 12, 2014

Supervisor:
Jörg Tiedemann, Uppsala University

Abstract

Machine transliteration is the process of automatically transforming the script of a word from a source language to a target language based on the pronunciation. It is a useful complement for a machine translation system, and important in cross-language information retrieval. In this thesis, focusing on the problem of transliterating names from different source languages into Chinese, we build a HMM based machine transliteration system. We use four different approaches to modify the baseline system. The modified system achieves ACC of 0.370 and Mean F-score of 0.714, which is close to the precision of the state-of-art machine transliteration systems. We apply the constructed systems on 14 different language sets for further experiments. The results show that the language specific systems are more accurate and efficient than the generic system. They are particularly effective on certain language sets. On the other hand, the language specific systems have some limitations, as each system is only oriented to a certain language. Our future work will include improving the accuracy of our general transliteration model as well as expanding and modifying the language specific transliteration systems.

Contents

List of Figures

List of Tables

Preface

First of all, I would like to show my great appreciation to Professor Jörg Tiedemann, who has supervised the thesis with great devotion, proposing constructive ideas and advice throughout the entire work, as well as providing actual experimental data and tools. I would also like to thank all the teachers of the Department of Linguistic and Philology at Uppsala University who helped me with my thesis and my studies during the master's program.

A special thanks must also go to Dr. Anders Wall. With his kind sponsorship, I was able to come and study in Sweden and obtain a master degree at Uppsala University. Thanks to all the people, including my friends, classmates and strangers who ever helped to make my life easier in Sweden.

Last but not least, I am very grateful to my parents. They are always encouraging and supporting me, far away from my homeland.

Best wishes to all of you.

1 Introduction

Along with the general development of natural language processing, the quality of machine translation has improved significantly. However, there are still many challenging problems remaining to be solved to improve machine translation systems further, for instance, the processing of unknown words, which are also known as out-of-vocabulary (OOV) words. Currently there are several successful commercial machine translation systems such as Google Translator and Bing Translator. Their solution is to copy the original form of the word in the source language directly in the generated translation. Considering the fact that a large number of the unknown words are named entities, this is actually a good option when the source and target languages have the same or similar alphabet, for example English and Swedish. But it is not ideal when the two languages have very different writing systems, for instance English and Chinese. In this case, it is better to provide transliteration instead.

First of all, we can see an example of translation outputs provided by Google Translator.

```
Source Sentence: Eric Saade is a Swedish singer.
Output(Swedish): Eric Saade är ett svenskt sångerska.
Output(Chinese): 埃里克Saade是瑞典歌手。
```

Figure 1.1: Example of an output by Google Translator

`Eric Saade` is the full name of the singer. His first name `Eric` is comparatively common and contained in the training data that are used to build the translation system. `Saade` is an uncommon surname which is an unknown word to the translation system. In the Chinese translation, we can see that `Eric` is translated correctly while `Saade` is not translated. Overall this is not an acceptable translation. Thus, an additional transliteration component will be very beneficial to the entire machine translation system. Furthermore, machine transliteration also plays an important role in cross-language information retrieval.

Machine transliteration is the process of automatically transforming the script of a word from a source language to a target language, while preserving pronunciation (Karimi et al., 2011). The machine transliteration problem concerns both transliteration detection and generation. In this thesis, we focus on transliteration generation of people's names, a subclass of named entities. Meanwhile we consider only the problem where Chinese is the target language.

In the first section, regarding transliteration as a labelling process, we build a baseline transliteration system using a hidden Markov model (HMM). We add Chinese Pinyin as intermediate representation based on the characteristics of the Chinese language, extending the classical HMM. We use the M2M Aligner to solve the segmentation problem and to obtain the alignment result for the translated segments. Similarly to most HMMs, we use the Viterbi algorithm for decoding. After the baseline system is established, ACC and mean F-score are used as two measures to evaluate the quality. Furthermore, based on the analysis of test results, we apply several approaches to improve the performance of the baseline system and build a modified system. The baseline system gets an ACC of 0.336 and mean F-score of 0.691 while the modified system improves them to 0.370 and 0.714. Later we apply both the baseline system and the modified system to the experiments on different language sets.

Transliteration of people's names greatly depends on the language origins since in most cases transliteration is phonetic-based. Names with identical written form in different languages can be pronounced differently and therefore the transliterations are also different. For example, "James" as an English name is transliterated as "詹姆斯" in Chinese while as a Spanish name it is transliterated as "哈梅斯" . In the second section of the thesis we therefore investigate how the language origins of the source names influence the quality of machine transliteration. We do experiments representatively with data sets representing Swedish, Turkish, English, German, Russian, French, Finnish, Portuguese, Spanish, Hungarian, Italian, Romanian, Serbian and Czech names. We test all the languages with both our baseline and the modified transliteration models. The results are compared and analyzed with the model trained without discrimination of language origins. The experiment results show that the language specific systems perform much better than the generic system, which indicates that discrimination of language source is important in machine transliteration.

We describe our works in this thesis thoroughly in the following chapters.

- In chapter 2, we review background research in both general machine transliteration and particularly where Chinese is used as the target language.
- In chapter 3, the data that are used for training and testing the transliteration system as well as the external tool M2M Aligner are introduced.
- In chapter 4, we describe how the baseline transliteration system is established using HMM and how we modify it with different approaches.
- In chapter 5, both the baseline and modified systems are applied to different language sets. We compare and analyze the results.
- In chapter 6, we summarize and conclude our work in this thesis.

2 Background

There are a number of publications on machine transliteration. In most cases, the source language and target language are English and an Asian language, for examples Chinese, Korean or Japanese. Similar to machine translation, machine transliteration also started with attempts to create rule-based systems. Later it turned out that due to the great flexibility of transliteration, it requires massive manual work to build complex rules to generate reasonable outputs. Thus, reasonable transliteration accuracy cannot be guaranteed. The research of machine transliteration greatly developed with the utilization of statistical models. Various phonetic-based and orthographic-based models devoted to different languages were built. A large number of approaches particularly focus on transliteration in which Chinese is used as the target language. Furthermore, there are standard metrics to evaluate the quality of automatic transliteration system, which are introduced in detail in section 2.4.

2.1 Phonetic-based Models

Transliteration is essentially phonetic-based. The very early study of automatic transliteration starts with Arbabi et al. (1994)'s research. They use a supervised artificial neural network (ANN) along with a knowledge-based system (KBS) to automate the transliteration process. The ANN filters out the unreliable candidates that will be inappropriately handled by the KBS, while the KBS uses linguistic rules for romanization.

A few years later, Knight and Graehl (1997) used weighted finite-state automata for Japanese Katakana to English back transliteration. The entire transliteration procedure is divided into five sub problems:

1. An English phrase is written.
2. A translator pronounces it in English.
3. The pronunciation is modified to fit the Japanese phonetic system.
4. The sounds are converted into Japanese Katakana.
5. The Japanese Katakana is written.

Using a generative model, all five probability distributions can be calculated respectively as:

- $\mathrm{p}(w)$ generates written English word sequences
- $\mathrm{p}(e|w)$ pronounces English word sequences

- $p(j|e)$ converts English sounds into Japanese sounds
- $p(k|j)$ converts Japanese sounds into Japanese Katakana
- $p(o|k)$ introduces misspellings caused by Optical Character Recognition (OCR)

$p(w)$ is implemented as a weighted finite state acceptor (WFSA), while the other conditional distributions are implemented as weighted finite state transducers (WFSTs).

Using Bayes' Rules, those probability distributions are recombined as:

$$p(w)\,p(e|w)\,p(j|e)\,p(k|j)\,p(o|k) \tag{2.1.1}$$

Given a katakana string o observed by OCR, the English word sequence w that maximizes the sum over all e, j, and k can be found.

Jung et al. (2000) apply an extended Markov window method to build the model for English to Korean transliteration. They use the pronunciation symbols for English words as defined in the Oxford computer usable dictionary (Roger, 1992) and construct English to Korean phonetic mapping tables that meet syllabification and alignment requirements for training the transliteration model. Based on the characteristics of the Korean language, they build a statistical tagging model by extending the Markov window with some mathematical approximation techniques. Their alignment and syllabification method improves transliteration accuracy as well as efficiency.

2.2 Orthographic-based Models

Recent studies show that the performance of orthographic-based models can be comparatively good as or even better when compared to phonetic-based alternatives. The direct orthographic mapping between the source and target languages can avoid some potential errors by eliminating a number of intermediate phonetic representations (Li et al., 2004).

Abduljaleel and Larkey (2003) build an n-gram transliteration model for English to Arabic transliteration. They present a simple statistical technique to train an English to Arabic transliteration model from pairs of names. They use proper name lists to train the n-gram model using GIZA++. The training procedure has two stages. First the model learns which n-gram segments should be added to the unigram inventory for the source language, and then in the second stage it learns the transliteration probabilities over this inventory. The model is a set of conditional probability distributions over Arabic characters, conditioned on English unigrams and selected n-grams. During transliteration, an English word S is first segmented according to the n-gram inventory, and for each segment, all possible Arabic transliterations T are generated. All the candidates are then scored and the best one is selected.

The adaptation of statistical machine translation for machine transliteration is also becoming popular and has proven to be fairly effective. The SMT system regards the transliteration problem as a normal machine translation task on character level. The IBM models as well as recent phrase-based SMT systems can be applied (Koehn et al., 2003). Compared to traditional machine translation,

there are slight differences. For instance, transliteration involves no reordering, which actually makes it easier.

Additionally, there are also some hybrid and correspondence-based transliteration models. According to the research of Oh et al. (2006), compared to elementary phonetic-based and orthographic-based models, hybrid and correspondence-based models are more effective. These models can work complementarily and hence generally improve the quality of machine transliteration.

2.3 Transliterations Models where Chinese is the Target Language

There are several difficulties in transliterating foreign names into Chinese. First, written Chinese is a logogram language. The phonetic representation of Chinese characters Pinyin is thus used as an intermediate romanization, which is important and has proved to be effective in machine transliteration, especially if using phonetic-based models. Second, each Chinese character is monosyllabic with a specific combination pattern of consonants and vowels. One Chinese character can be used to transliterate a complete syllable or just a consonant. For example, /eks/ is one syllable in English but needs three Chinese characters to transliterate, which are three syllables in Chinese (/e/ + /ke/ + /si/). Thus, English syllables cannot easily be mapped to Chinese characters nor pinyin accurately.(Zhou, 2009)

There is a large portion of machine transliteration studies in which Chinese is used as the target language. Li et al. (2004) present a transliteration framework that allows direct orthographical mapping (DOM) between English and Chinese. They use an n-gram transliteration model (TM) to derive the aligned transliteration units from a bilingual dictionary. They use the Expectation–Maximization (EM) algorithm for the transliteration alignment and the estimation of the relevant transliteration probabilities. In contrast to the noisy-channel model, the TM does not try to capture how source names can be mapped to target names, but rather how source and target names can be generated simultaneously. Chen et al. (2011) extended the joint source channel model into a multi-to-multi joint source-channel model, which allows alignments between substrings of arbitrary lengths in both source and target strings. Their model improves significantly on the original when integrated into a modified phrase-based statistical machine translation system.

Zhou (2009) proposes an English-Chinese name transliteration system using a maximum n-gram Hidden Markov Model (HMM). They apply a two-phase transliteration model by building two HMM models, one between English and Chinese Pinyin and another between Chinese Pinyin and Chinese characters. Their model improves traditional HMM by assigning the largest weight to the longest prior translation sequence of syllables. They also use a web-mining module as an external tool to improve the performance by adding online popularity information for candidate translations. They implement a syllabification process for mapping the transliterated fragments.

Kwong (2009) develops a transliteration system with sound model and tone model based on the characteristics of the Chinese language. As Chinese is a tone

language, the tone patterns are cognitive in nature. Some combinations may just sound strange. They develop a sound model (SoM) and a tone model (ToM) as a post evaluation procedure for Li et al. (2004)'s Joint Source-Channel Model. SoM basically assembles the homophones and captures the sound patterns in terms of a grapheme-phoneme mapping. ToM captures the tone patterns of transliteration, irrespective of the sound and the character choice.

2.4 Evaluation of Machine Transliteration

In the NEWS Proceedings of the Named Entities Workshop: Shared Task on Transliteration, there are 4 different metrics for measuring the quality of machine transliteration systems via the submitted results. For a submitted input list, up to 10 different candidate transliterations can be accepted. (Li et al., 2010b)

The following notations are further assumed:

- N: Total number of names (source words) in the test set
- $r_{i,j}$: $j-th$ reference transliteration for $i-th$ name in the test set
- K_i: Number of candidate transliterations produced by a transliteration system

`1. Word Accuracy in Top-1 (ACC)`, also known as Word Error Rate, it measures correctness of the first transliteration candidate in the candidate list produced by a transliteration system. $ACC = 1$ means that all top candidates are correct transliterations. In other words, they match one of the references. $ACC = 0$ means that none of the top candidates are correct.

$$ACC = \frac{1}{N}\sum_{i=1}^{N}\left(\begin{cases}1, & \text{if } \exists r_{i,j} : r_{i,j} = c_{i,1}\\ 0, & \text{otherwise}\end{cases}\right) \tag{2.4.1}$$

`2. Mean F-score`, also mentioned as Fuzziness in Top-1, which measures how different on average the top transliteration candidate is from its closest reference. F-score for each source word is a function of Precision and Recall and equals 1 when the top candidate matches one of the references and 0 when there are no common characters between the candidate and any of the references.

Precision and Recall are calculated based on the length of the longest common subsequence between a candidate and a reference:

$$LCS(c, r) = \frac{1}{2}(|c| + |r| - ED(c, r)) \tag{2.4.2}$$

where ED is the edit distance and $|x|$ is the length of x. For example, the longest common subsequence between "abcd" and "afcde" is "acd" and its length is 3. The best matching reference, that is, the reference for which the edit distance has the minimum, is taken for calculation. If the best matching reference is given by

$$r_{i,m} = \arg\min_{j}\left(ED\left(c_{i,1}, r_{i,j}\right)\right) \tag{2.4.3}$$

then Recall, Precision and F-score for $i-th$ word are calculated as

$$R_i = \frac{LCS\left(c_{i,1}, r_{i,m}\right)}{|r_{i,m}|} \tag{2.4.4}$$

$$P_i = \frac{LCS\left(c_{i,1}, r_{i,m}\right)}{|c_{i,1}|} \tag{2.4.5}$$

$$F_i = 2\frac{R_i \times P_i}{R_i + P_i} \tag{2.4.6}$$

The length is computed in distinct Unicode characters.

`3. Mean Reciprocal Rank (MRR)` Measures traditional MRR for any right answer produced by the system, from among the candidates. 1/MRR tells approximately the average rank of the correct transliteration. MRR closer to 1 implies that the correct answer is mostly produced close to the top of the n-best lists.

`4. MAPref` Measures tightly the precision in the n-best candidates for i-th source name, for which reference transliterations are available. If all of the references are produced, then the MAP is 1.

If the transliteration system only returns one candidate transliteration, we refer only to ACC and Mean F-Score to observe its performance. In addition, there is an official script provided to calculate those measures, which is used in this thesis.

3 Data and Tools

To train and test a machine transliteration system, we need large amount of parallel corpora which in our case only contains names. In this thesis, we use the dataset of NEWS 2010 shared task (Li et al., 2010b), specifically English to Chinese generation task to build and test our basic transliteration model in order to compare it with those state-of-art transliteration systems evaluated in the conference. Meanwhile, the Translation Dictionary for Foreign Names (Xia, 1993) is used as the main data set for investigating how different language origins of the names influence the quality of machine transliteration system. We also use a Chinese character-pinyin dictionary to get the pinyin of transliterations as pinyin is used as intermediate of our transliteration system. Additionally, as the crucial tool for retrieving the segmentations of the source string and getting the alignment substring pairs, M2M Aligner is also introduced.

3.1 Data Set of NEWS 2010 Shared Task

The Named Entity Work Shop 2010 provides standard training and testing data sets for the machine transliteration generation shared task. The corpora are in XML format and encoded in UTF8. The major part of the data is derived from the data that were used in NEWS 2009. (Li et al., 2010a)

The training set for English to Chinese transliteration contains 31,961 terms. All terms have only one reference transliteration. The original development set contains 5,792 terms and it is in the same format as the training set. The original testing set contains 3,000 terms without the reference transliterations, which cannot be used in our experiment.

In this thesis, we split the original development set into two sets which both contain 2,896 terms. We use one as our development set and the other as our final test set.

3.2 Translation Dictionary for Foreign Names

Translation Dictionary for Foreign Names[1] (Chinese: 《世界人名翻译大辞典》) is a reference book for manual translation for Chinese. It contains around 650,000 different names as well as their transliterations and source language, covering more than 100 language origins. Additionally, some supporting rules for manual transliteration are included in case the queried term is not contained

[1]This is not the official English name of the dictionary. This translation is given by the author of the thesis.

in the dictionary. The electronic version of the dictionary is available on the internet. Figure 3.1 is a sample of the dictionary.

Aab	Swedish	奥布
Aagaard	Swedish, Danish, Norwegian	奥高
Ärynen	Finnish	埃吕宁
Egilsson	Icelandic	埃吉尔松
Eglentyne	?	埃格伦蒂娜
Hamodi	Arabic	哈穆迪(男名)
Primrose	English	普里姆罗斯;普丽姆罗丝
Sarafannikov	Russian	萨拉凡尼科夫
Prijović	Serbian	普里约维奇
Aakjaer, Jeppe	Danish	奥克亚<丹>诗人、小说家。
Zentoku	Japanese	全德(姓)

Figure 3.1: Sample of Translation Dictionary for Foreign Names

We can see that Arabic and Russian are already transcribed into the classical Latin alphabet according to English pronunciation. Other languages that use the Latin alphabet all keep their own particular letters and annotations that do not exist in English. Unfortunately, we cannot use the dictionary directly as training and testing data for some of the terms are not in the strict {term}-{language source}-{transliteration} pattern. Moreover, some terms contain redundant information while some terms do not state the language source. Additionally, all terms with different language sources are mixed together. Some preprocessing is therefore required for the dictionary.

- First of all, we fix the encoding problems of the dictionary to make sure that all the letters in different languages can be shown properly.
- We remove those terms without language source information, like the term “Eglentyne”.
- We also use a special pattern to take away the proprietary terms that contain irrelevant information which is detrimental to our transliteration system but difficult to filter out. For example, the term “Aakjaer, Jeppe”, in the Chinese text, “奥克亚” is just the transliteration of “Aakjaer” while “丹” is his nationality and “诗人、小说家” is his occupation. These terms normally contain commas in the source text and some particular text marks like “<”, “>”and “、”in the target text.
- For those with more than one transliteration, for example the term “Primrose”, with two transliterations “普里姆罗斯” and “普丽姆罗丝”, we keep just the first one and abandon the rest. These terms are only a small portion of the data set; eliminating the extra transliteration does not really make a difference.
- For the names that share the same transliteration in different languages as the term “Aagaard”, we regard them as different terms respectively with all the language sources.

Languages	Numbers of the Terms
Czech	35,767
English	50,267
Finnish	18,286
French	83,970
German	51,246
Hungarian	28,445
Italian	61,174
Portuguese	10,713
Romanian	29,944
Russian	50,276
Serbian	35,053
Spanish	30,667
Swedish	30,753
Turkish	15,121

Table 3.1: 14 selected subsets and the numbers of their terms contained

The modified terms are grouped into subsets according to their language sources. To ensure that the transliteration model can be trained with sufficient data, we select those subsets which contain more than 10,000 terms for further experiments. In addition, as the translation of Asian names, such as Japanese, Korean and Vietnamese is not regular transliteration in the same sense as the others due to their special relations with Chinese, these sets are also omitted. This leads to 14 selected subsets, whose detailed information is shown in Table 3.1.

3.3 Chinese Character-pinyin Dictionary

If a Chinese character is given, its pinyin is deterministic. Usually there is exactly one unique romanization for any given Chinese character. Considering the fact that the Chinese characters used for transliteration are very limited, we can use a Chinese character-pinyin dictionary to get the pinyin that we need to building the transliteration system. Figure 3.2 shows a sample of the dictionary that we use in this thesis. Both the romanization and tones are included.

3.4 M2M-Aligner

Many-to-many Aligner is the implementation of the many-to-many alignment algorithm proposed by Jiampojamarn et al. (2007). This algorithm finds the lexicon alignments without using any annotated training data or extra linguistic knowledge. The training of the many-to-many aligner is an extension of the forward-backward training of the one-to-one stochastic transducer originally presented by Ristad and Yianilos (1998), which essentially uses an Expectation–Maximization (EM) algorithm. M2M Aligner also uses a bigram letter chunking prediction that automatically discovers and processes double letter

白	bai2
柏	bai3
柏	bo2
柏	bo4
百	bai3
百	bo2
摆	bai3
佰	bai3
败	bai4

Figure 3.2: Sample of Chinese character-pinyin Dictionary

combinations. M2M aligner has been applied in the research of letter-to-phoneme conversion as well as name transliteration.

M2M reads the input in two formats "l2p" and "news". In this thesis, we preprocess our data according to the "news" format. Each token is separated by a space, a tab separates between source and target strings that are in the same line. Figure 3.3 shows a sample of input data, where source strings are English source names and target strings are Chinese pinyin of their transliterations.

a b a d i	a ba di
a b a d y	a ba di
a b a g a i l	a bi gai er
a b a r b a n e l	a ba bai nei er
a b a r e	a bei er
a b a s t e n i a	a ba si di ni ya
a b b e l	a bei er

Figure 3.3: Sample of input of M2M Aligner

a|b:a|d:i|	a|ba|di|
a|b:a|d:y|	a|ba|di|
a|b|a:g:a:i|l|	a|bi|gai|er|
a|b:a|r:b:a|n:e|l|	a|ba|bai|nei|er|
a|b|a:r:e|	a|bei|er|
a|b:a|s|t:e|n:i|a|	a|ba|si|di|ni|ya|
a:b|b:e|l|	a|bei|er|

Figure 3.4: Sample of output of M2M Aligner

There are a number of settable parameters. "–maxX" and "–maxY" are the maximum lengths of substrings in the source and target strings. Apart from these two, we use default settings for all the other parameters when doing experiments in this thesis. Figure 3.4 is a sample of output by M2M Aligner, we are able to get the segmentations as well as alignments that are required for constructing our transliteration system. However, we should notice that there

are some mistakes in the output. For example, the correct alignment result of “a b b e l“ and “a bei er” should be “a|b:b:e|l| a|bei|er|“ instead.

In order to have some knowledge about the M2M Aligner’s performance, we randomly select 20 instances from the output and manually evaluate the precision, which is calculated by 3.4.1. We get a fairly low precision of 0.518, which implies that we have many alignment errors that are detrimental when used as training instances for our transliteration system.

$$\text{Precision} = \frac{\text{Number of correct alignments}}{\text{Total number of alignments}} \tag{3.4.1}$$

4 Construction of Machine Transliteration System

4.1 Baseline System

The aim of transliteration system is to return the candidate target transliteration result with the highest probability within the system given the source string. We use formula 4.1.1 to represent this process, where t is the target transliteration and s is the source string.

$$t^* = \arg\max_t P(t|s) \tag{4.1.1}$$

In this thesis, we regard transliteration as a labelling process, similar to part-of-speech tagging. We assume that the source string is constructed of several substrings. Each of the substrings performs as a unit which is able to map to specific target strings in transliteration, which here are the Chinese characters. We use σ to present the segmentation of the source string s.

$$\arg\max_t P(t|s) = \arg\max_t \sum_\sigma P(t|\sigma)P(\sigma|s) \tag{4.1.2}$$

$$\arg\max_t P(t|\sigma) = \arg\max_t P(t_1t_2.....t_n|\sigma_1\sigma_2.....\sigma_n) \tag{4.1.3}$$

Considering the dependencies between the substrings, we can apply the Hidden Markov Model (Ghahramani, 2001) to build the system. The substrings of the source name make up the observed sequence. Via the HMM model we estimate the hidden sequence, which is the target transliteration.

4.1.1 Using Pinyin as Intermediate

Instead of implementing a direct mapping between the source string and Chinese characters, we use pinyin, the romanization of Chinese characters, as the intermediate. Pinyin as an additional feature in transliteration was first introduced by Li et al. (2004) and has proven to be effective in subsequent studies. Compared to the set of Chinese characters that are used in transliteration, the set of the corresponding pinyin is smaller as several Chinese characters may share the same pronunciation. Moreover, the transliteration process is phonetic based, and Chinese characters essentially do not represent pronunciations.

As we know, there is a dependency between Chinese characters and their pinyin. For the training data, given the target transliterations in form of characters we get the pinyin via the Chinese character-pinyin dictionary. Some

characters are polyphones and therefore have several corresponding pinyin in the dictionary; we take the first and disregard the rest. Furthermore, we omit the tone information contained in the dictionary as most source languages are not tone-languages. We regard tone as a irrelevant feature in transliteration, which is different from Kwong (2009)'s research.

After retrieving pinyin, the training data actually contains three parts: source string, pinyin and target string. The transliteration process then contains two mapping phases. First is from source string to pinyin, then from pinyin to target string. The model is described by 4.1.4 when pinyin is added as the intermediate, where r stands for pinyin, the romanization form of Chinese characters.

$$\arg\max_{t} P(t|\sigma) = \arg\max_{t} P(t|r)P(r|\sigma) \tag{4.1.4}$$

4.1.2 Segmentation

The existing training data are not segmented. We need to do the initial segmentation in order to build the HMM model. The quality of the segmentation has a significant impact on overall performance of transliteration system. Unfortunately we do not have existing manually annotated data to train a supervised segmentation model.

In this thesis, we use the M2M Aligner which segments the strings on both source and target sides as well as provides the alignments of the segmented substrings for the training data. We only use the aligner on the first mapping phase, from source string to pinyin. Because of the dependency between pinyin and Chinese characters, the segmentation and alignment of the second mapping phase is already determined. In the baseline model, we set the maximum length on the source side as five and on the target side as one considering one single Chinese character is normally an independent phonetic unit. This actually might be problematic. For example, the source name "Max",

Ma| x ma| ke si 马|克思

where the letter x is transliterated into two Chinese characters. However, this is also the only case in English that one single letter from the source string is transliterated into two strings on the target side. A good solution is to regard letter x as an exception, which will be discussed in the next section on modified systems.

For a certain generated transliteration t, theoretically we should sum up all the probabilities with different segmentations. In this case, we would have to compute the probabilities of all possible outputs. In order to improve efficiency, for each specific output we only consider one possible segmentation σ^*, which maximizes the overall probability. We assume that for this specific output t, the segmentation probability of σ is much greater than the other segmentations σ', as $P(\sigma|s) \gg P(\sigma'|s)$. This makes the model 4.1.5.

$$\arg\max_{t} P(t|s) \approx \arg\max_{t} P(t|r)P(r|\sigma)P(\sigma|s) \tag{4.1.5}$$

4.1.3 Two-phase Hidden Markov Model

According to Bayesian chain rule, we can rewrite some probabilities as in 4.1.6.

$$\begin{aligned} P(r|\sigma) &= \frac{P(r,\sigma)}{P(\sigma)} = \frac{P(r)P(\sigma|r)}{P(\sigma)} \\ P(t|r) &= \frac{P(t,r)}{P(r)} = \frac{P(t)P(r|t)}{P(r)} \\ P(t|r)P(r|\sigma) &= \frac{P(t)P(r|t)}{P(r)} \cdot \frac{P(r)P(\sigma|r)}{P(\sigma)} = \frac{P(t)P(r|t)P(\sigma|r)}{P(\sigma)} \end{aligned} \tag{4.1.6}$$

In the transliteration task, the source string s is given and we segmented it, which makes the formula as in 4.1.7.

$$P(t|r)P(r|\sigma) \propto P(t)P(r|t)P(\sigma|r) \tag{4.1.7}$$

For $P(r|t)$, as we know that given the Chinese characters, the Romanization is deterministic, so actually $P(r|t) = 1$ because for all invalid transliterations r', $P(r'|t) = 0$. Then the equation becomes 4.1.8.

$$P(t|r)P(r|\sigma) \propto P(t)P(\sigma|r) \tag{4.1.8}$$

For $P(t)$ and $P(\sigma|r)$ in 4.1.8,

$$P(t) = P(t_1|t_0)P(t_2|t_0,t_1).....P(t_n|t_0,t_1.....t_{n-1}) \tag{4.1.9}$$

$$P(\sigma|r) = P(\sigma_1|r_1)P(\sigma_2|r_1,r_2,\sigma_1).....P(\sigma_n|r_1...r_n,\sigma_1...\sigma_{n-1}) \tag{4.1.10}$$

Here t_0 is the additional mark for the beginning of a string. Based on Markov assumption, we apply the first-order model and simplify $P(t)$ in our baseline model as in 4.1.11.

$$P(t) \approx P(t_1|t_0)P(t_2|t_1).....P(t_n|t_{n-1}) \tag{4.1.11}$$

Furthermore, we assume that each sequence σ_i only depends on the current corresponding r_i, which makes $P(\sigma|r)$ as in 4.1.12.

$$P(\sigma|r) \approx P(\sigma_1|r_1)P(\sigma_2|r_2).....P(\sigma_n|r_n) \tag{4.1.12}$$

Now we can integrate everything together in 4.1.13. However, we still need to add the segmentation probability to estimate $P(t|s)$, which will be introduced in next section.

$$\arg\max_t P(t|\sigma) = \arg\max_t \prod_i P(t_i|t_{i-1})P(\sigma_i|r_i) \tag{4.1.13}$$

4.1.4 Training

We use maximum likelihood estimation with the result returned by M2M Aligner to estimate the parameters of the model described in the previous section.

From the result, we can get all the possible substrings, their frequencies as well as their aligned pinyin. First, we use a unigram model to estimate the

segmentation probability of the source string. We assume that the substrings σ are independent on s. In this case, for a given source string s and the most probable segmentation σ for transliteration,

$$\begin{aligned} P(\sigma|s) &= \frac{P(\sigma, s)}{P(s)} \\ P(\sigma, s) &= P(\sigma) \end{aligned} \tag{4.1.14}$$

so we can estimate the segmentation probability as 4.1.15.

$$\arg\max_{\sigma} P(\sigma|s) = \arg\max_{\sigma} P(\sigma) \tag{4.1.15}$$

The segmentation probability is then computed using unsmoothed Maximum Likelihood Estimation(MLE) as in 4.1.16, where $f(s_i)$ is the frequency of the substring in the alignment result and N is the total number of all substrings.

$$\arg\max_{\sigma} P(\sigma) = \prod_{i=1}^{n} P_s(\sigma_i) = \prod_{i=1}^{n} \frac{f(\sigma_i)}{N} \tag{4.1.16}$$

$P(t_i|t_{i-1})$ is the probability of target string t_i appearing after t_{i-1}. Once more, we use MLE to estimate the probability by 4.1.17, where $f(t_{i-1}, t_i)$ is the number of times t_i appears after t_{i-1} in the target Chinese transliterations in the training data and $f(t_{i-1})$ is the total frequency of t_{i-1}.

$$P(t_i|t_{i-1}) = \frac{f(t_{i-1}, t_i)}{f(t_{i-1})} \tag{4.1.17}$$

$P(\sigma_i|r_i)$ is the probability of the source string given the corresponding pinyin. We estimate it by 4.1.18, where $f(r_i, \sigma_i)$ is the number of times that σ_i is aligned to r_i and $f(r_i)$ is the total number of r_i.

$$P(\sigma_i|r_i) = \frac{f(r_i, \sigma_i)}{f(r_i)} \tag{4.1.18}$$

In summary, the probability of the complete model can be computed by 4.1.19. We can see that we integrate all the probability estimates together, including the segmentation probability. The integrated model ensures that we find the segmentation $P(\sigma)$ that maximizes the final and entire probability of the transliteration model. Otherwise if we did the segmentation process separately, the segmentation $P(\sigma')$ that we would get only maximizes the segmentation model, which may not be optimal for the complete transliteration model.

$$\arg\max_{t} P(t|s) \propto \arg\max_{t} \prod_{i=1}^{n} \frac{f(\sigma_i) f(t_i - 1, t_i) f(r_i, \sigma_i)}{N f(t_{i-1}) f(r_i)} \tag{4.1.19}$$

Furthermore, using this procedure, our model actually visibly becomes a Markov model, trained with the segmentation and alignment information provided by M2M Aligner.

4.1.5 Decoding

The complexity of a naive decoding algorithm becomes exponential if we simply calculate the probabilities of all the possible outputs as in 4.1.19, which is definitely unacceptable efficiently. However, the HMM model makes it possible to find the best transliteration path which gets the maximum likelihood according to the model. The most common decoding algorithm for HMM is Viterbi algorithm, which is a kind of dynamic programming (Forney, 1973).

Following the notation of Jurafsky and Martin (2008), for the standard Viterbi Algorithm, given state q_k at time i, the value $v_i(l)$ is computed by 4.1.20.

$$v_i(l) = \max_{k=1}^{N} v_{i-1}(k) a_{kl} b_i(o_i) \tag{4.1.20}$$

$v_{i-1}(k)$ is the previous Viterbi path probability from the previous time step. a_{kl} is the transition probability from previous state q_k to current state q_1. $b_1(o_i)$ is state observation likelihood of the observation symbol o_i given the current state l. In our case, $v_{i-1}(k)$ is the cumulative probability of the previous transliterated sequence before current term. a_{kl} is $P(t_l|t_k)$, the probability of the character in the current term appearing after the character in the previous term. $b_l(o_i)$ is $P(\sigma_l|r_l)P(\sigma_l)$, the transliteration probability of the substring multiplied by its segmentation probability.

In our approach, we first go over the source string and check all the substrings that are no longer than five. If the substring appears in the result returned by M2M aligner, we add it to a candidate $\mathtt{listA}$. For each substring in the candidate $\mathtt{listA}$, according to its length and position we can get all its previous substrings. We can also find those substrings that are at the end of the source string.

For all the substrings in the candidate $\mathtt{listA}$, we find all the possible pinyin that they are aligned to in the training set. Then through pinyin we can get all the possible corresponding Chinese characters. We build candidate $\mathtt{listB}$ based on these different characters, with which the Viterbi Algorithm will be performed. For a term X in candidate $\mathtt{listB}$, it contains the following information:

1. A unique identification number ID in the list.
2. Cumulative probability P from the previous steps.
3. Substring σ_i from the original source string.
4. Pinyin r_i that σ_i is aligned to.
5. Chinese character t_i that r_i is mapped to.
6. Segmentation probability $P_s(\sigma_i)$ of the substring σ_i.
7. Transliteration probability of σ_i given r_i, which is $P(\sigma_i|r_i)$.
8. The identify numbers Pre_{IDs} of the terms whose substrings are the previous substrings of σ_i. For convenience, we mention those terms as previous terms of current term X.

9. Back pointer $-\mathtt{Best}_{\mathtt{Pre}}$ that records the ID of the best previous term which makes the current term get the maximum cumulative probability.
10. Index of σ_i in the source string, $\mathtt{IN}_S$.
11. Whether the substring σ_i is the end of the source string, $\mathtt{Ends}_S$.

Initialization

First we sort the terms in candidate listB by their indexes in the source string to make sure that we compute the substrings that are closer to the beginning earlier. For a term in the list, if the substring is at the beginning of the source string, it has no previous substrings. We use these terms with starting substrings to set the initial state of Viterbi algorithm as 4.1.21.

$$\nu_l(l) = P(t_l|t_0)P(\sigma_l|r_l)P(\sigma_i) \tag{4.1.21}$$

Recursion

All the other terms have previous terms. We can use recursion to compute all the probabilities. As in 4.1.22, for each of the previous terms Pre, we compute the cumulative probability with the current term. The one that obtains the highest probability is the current term's best previous term and the probability is assigned as current term's cumulative probability. Meanwhile we also record its ID in the back pointer $-\mathtt{Best}_{\mathtt{Pre}}$.

$$\begin{aligned} \nu_i(l) &= \max_{k=1}^{N} \nu_{i-1}(k) a_{kl} b_l(o_i) \\ b\nu_i(l) &= \arg\max_{k=1}^{N} \nu_{i-1}(k) a_{kl} b - l(o_i) \end{aligned} \tag{4.1.22}$$

Termination

After the cumulative probabilities of all terms in the candidate listB are computed, we find all the terms whose substrings are at the end of the original source string. Those terms are the terminations of the Viterbi path. We compare their probabilities to get the one who has the highest probability, which records the best Viterbi path in the search space. Following the back pointer $-\mathtt{Best}_{\mathtt{Pre}}$, we can back trace all the terms in the path and get the Chinese characters from the end to the beginning. The best output of the model is then generated.

The decoding algorithm is illustrated in 4.1.

4.1.6 Evaluation of Baseline System

We use the data set of NEWS 2010 Shared Task to evaluate our baseline system. We use the training set to build our model and then test it using both the development set and testing set. The ACC and Mean F-Score of the baseline model are shown in Table 4.1. Compared to the two systems that are submitted in NEWS 2010, our baseline model achieves comparatively high ACC and Mean F-score. Detailed information on the two submitted systems will be introduced in 4.2.5.

```
create candidate listA;
create candidate listB;
sort listB by IN_S;
foreach term X in listB do
    if X.Pre_IDs is empty then
        Initialization
    else
        Recursion
    end
end
find ∀T that T.End_S = True;
find T_max in T with highest cumulative probability P;
back trace and generate the output;
```

Figure 4.1: Basic structure of the decoding algorithm

	development set	testing set
ACC	0.352	0.336
Mean F-Score	0.692	0.691

Table 4.1: Performance of baseline system

4.2 Modified System

We apply different approaches to improve the performance of our baseline system. We focus on both increasing the precision of the external tool M2M Aligner as well as adjusting the internal parameters of the transliteration system. We test with the development set of NEWS 2010 shared task to see whether the approach that we apply is effective. Finally, we integrate all the positive approaches together to build our modified system.

4.2.1 Pre-contract Letter Combinations for M2M Aligner

The accuracy of the M2M Aligner is significant as errors in segmentations and alignments are very detrimental to our transliteration system. Our system should be improved if we reduce the errors committed by M2M Aligner.

As previously stated, M2M Aligner has a bigram chunking prediction part to process the letter combinations, while we can see from Figure 3.4 that there are still some mistakes that can be avoided if the letter combinations are handled properly. In our transliteration task, there are some letters on the source side that are pronounced as one single letter and are never transliterated into two Chinese characters, for example, the double "b" in "abbel". If we contract these two "b"'s together as one letter and then apply the data to the aligner, the segmentation and alignment result we get is:

a|bb:e|l| a|bei|er|

This is the correct result that we need. In this case, if we pre-contract all these letter combinations based on our linguistic knowledge, the accuracy of the M2M Aligner should be improved.

However, we cannot just simply contract all the common letter combinations in our system because some combinations are pronounced differently in different context. They may be transliterated into two Chinese characters on certain occasions. For example,

A|l|t|hou|se ao|er|te|hao|si| 奥|尔|特|豪|斯

“th”is a common combination and in most cases it is pronounced and transliterated as one unit. But in “Althouse”, which is a compound of "Alt" and "house", “t”is transliterated as “特”while “h”together with “ou”are aligned and transliterated as “豪”.

Moreover, letter combinations in different languages are not the same. We should specify different contract rules for different language sets. We apply one universal rule to all the language sets: if two identical letters appear continuously, they are contracted. As well as this, the particular letter combinations for different language sets are shown in Table 4.2. The contract rule for English is used when testing on NEWS 2010 data.

Languages	Letter Combinations
Czech	sz, zs, cs, dž
English	ch, ck, sh
Finnish	-
French	qu, ch
German	ch
Hungarian	dz, zs
Italian	ch, gh
Portuguese	ch
Romanian	gh, ch
Russian	ch, sh
Serbian	ch
Spanish	qu, ch
Swedish	ck, sj, hj
Turkish	-

Table 4.2: Letter combinations applied to different language sets

When we pre-contract the letter combinations in our training set and build our transliteration system, we get ACC of 0.372 and Mean F-score of 0.705 on our development set of NEWS 2010 data. Applying letter combinations helps the transliteration system perform better.

4.2.2 Trim Low Frequent Alignments

Before using the output of M2M Aligner to train our model, we notice that there are a number of alignments that only appear once. Statistically speaking, these low frequent terms are much less reliable than the high frequent ones.

In our transliteration task, we assume that those low frequent alignments are more likely to be incorrect alignments which are detrimental to the system. Trimming those terms largely decreases the entire search space, which makes our system more efficient.

However, we have to be careful when we apply this approach to smaller training sets. In some languages there are several very rare letters, which have very few alignments in M2M Aligner's output. If we regard them as low frequent alignments and discard them, it is very possible that those letters are entirely eliminated from the training data so that they become unknown tokens that cannot be handled by the decoding system.

Eventually, based on experiments on the development data set, we only trim those terms whose frequencies are 1. We then achieve an ACC of 0.361 and Mean F-score of 0.700.

4.2.3 Add Penalty Score

As introduced in section 4.1.4, we use a unigram model to estimate the segmentation probability. One of the problems of the unigram model is that the segmentation with longer substrings tends to have a higher probability. Fine-grained segmentations are penalized more strongly than coarse-grained ones. As stated in formula 4.1.16, the probabilities of all substrings are multiplied. We can see from 4.2.1 that when calculating the probability of a segmentation, the one with fewer substrings is divided by a smaller N^n, where n is the number of substrings and N is a constant, while n is smaller when those substrings are longer.

$$P(\sigma) = \prod_{i=1}^{n} \frac{f(\sigma_i)}{N} = \frac{f(\sigma_1)f(\sigma_2).....f(\sigma_n)}{N^n} \tag{4.2.1}$$

In order to compensate for this negative effect of the unigram model, as shown in 4.2.2, we add a penalty score $Penalty_1(\sigma_i)$ to modify the original probabilities of the substrings $P_s(\sigma_i)$ into $P_s^{'}(\sigma_i)$ to make the length of substrings have less impact on the overall estimation.

$$\begin{aligned} P_s^{'}(\sigma_i) &= P_s(\sigma_i) \times Penalty_1(\sigma_i) \\ Penalty_1(\sigma_i) &= (Len(\sigma_i))^{-\alpha} \end{aligned} \tag{4.2.2}$$

Here $Len(\sigma_i)$ is the length of substring σ_i and α is a constant.

In section 2.3 we stated that the syllable of Chinese characters has a specific pattern, which means some types of syllable in the source names cannot be transliterated by just one single Chinese character. In this thesis, we define an identification pattern IP as a penalty system shown in 4.2.3 to examine whether a substring can be transliterated into one single Chinese character.

$$IP = [vowel(s)][consonant(s)][vowel(s)] \tag{4.2.3}$$

If any part of σ_i fits IP, it means that σ is very unlikely to be transliterated into one character. Thus, we introduce another penalty score $Penalty_2(\sigma_i)$ as

in 4.2.4 to convert the original $P_s(\sigma_i)$ into a smaller $P_s^{'}(\sigma_i)$. β is a constant that $0 < \beta < 1$.

$$P_s^{'}(\sigma_i) = P_s(\sigma_i) \times Penalty_2(\sigma_i)$$
$$Penalty_2(\sigma_i) = \begin{cases} \beta & \text{if any part of } \sigma \text{ fits IP} \\ 1 & \text{otherwise} \end{cases} \tag{4.2.4}$$

Now we integrate the two penalty scores together as

$$P_s^{'}(\sigma_i) = P_s(\sigma_i) \times Penalty_1(\sigma_i) \times Penalty_2(\sigma_i) \tag{4.2.5}$$

Once again we experiment on the development set. The transliteration system performs best when $\alpha = 3$ and $\beta = 0.0001$, which obtains an ACC of 0.371 and Mean F-Score of 0.706.

4.2.4 Preprocess Letter "x"

From the previous section 4.1.2 we know that the letter "x" is the only case that one letter in the source name is aligned and transliterated into two Chinese characters. In English, in most cases "x" is pronounced as /ks/. However, in a few cases it is pronounced as one single unit therefore only transliterated by one character. For example,

Te|xey|ra　　te|xie|la　　特|谢|拉

This appears in the data set of NEWS 2010, although technically it is a Portuguese name. Apart from English, "x" is always pronounced as /ks/ in most European languages such as German, Swedish and Finnish. The situation is a bit different in Spanish and Portuguese. In Spanish, "x" is pronounced as /h/ on many occasions such as in "México" and "Xavi". But when "x" appears at the beginning of a name, it is always pronounced as /h/. It is even more complicated in Portuguese. When "x" is pronounced as one phonetic unit, it can be either /h/ or /ʃ/. However, similarly to Spanish, when "x" is the start of a name it is always pronounced as one single unit. In French, "x" is common and normally pronounced as /ks/ but silenced when it is at the end. The letter "x" is not common in some languages like Hungarian, and it does not even exist in Turkish.

In this thesis, we use a simple solution to the problem. We replace the letter "x" by "ks" in the source names before the decoding system transliterates them. For Spanish and Portuguese, we do not replace it if it is located at the beginning of the source name. Even though it is not always the case that "x" is pronounced similarly to the combination of "ks", our system still handles the exceptions in some way as we allow multiple letters from the source side to be aligned to one Chinese character. Thus, it is still possible that "x" will be transliterated correctly eventually.

The ACC and Mean F-score that we get on the development set after applying this replacing rule are 0.355 and 0.692.

4.2.5 Evaluation of Modified System

Table 4.3 shows the respective performances of the four modifying approaches that we tested on the development set of NEWS 2010 data in previous sections. We can see that Approach 4.2.1 is most effective in improving ACC while Approach 4.2.3 is best at increasing Mean F-score. Approach 4.2.4 does not appear to be as effective as the others superficially. But we should be aware that it is only oriented to the instances that contain the letter "x", hence we still consider it to be a significant boost to the transliteration system.

Modifying Approaches	ACC	Mean F-score
4.2.1	0.372	0.705
4.2.2	0.361	0.700
4.2.3	0.371	0.706
4.2.4	0.355	0.692

Table 4.3: Effectiveness of the Applied Modifying Approaches

We combine the four approaches to build our final modified transliteration system. Similarly to the baseline system, we evaluate our modified system with both the development and testing sets of NEWS 2010 data. The results are contained in Table 4.4. Compared to the baseline system, as shown in Table 4.5, our modified model achieves significant improvements.

	development set	testing set
ACC	0.387	0.370
Mean F-Score	0.718	0.714

Table 4.4: Performance of modified system

	development set (%)	testing set (%)
ACC	+0.0359 (10.21%)	+0.0335 (9.97%)
Mean F-Score	+0.026 (3.76%)	+0.023 (3.23%)

Table 4.5: Improvement of modified system compared to baseline system

Compared to the two English-Chinese transliteration systems that are submitted in NEWS 2010, our modified system performs slightly better than University of Alberta's system (Jiampojamarn et al., 2010). They achieve a ACC of 0.363 and a Mean F-score of 0.707. However, our modified system is still significantly worse than City University of Hong Kong's system (Chen et al., 2010). They use a SMT based transliteration model as the baseline, which gets a ACC of 0.381. They focus on improving the precision by reranking the multiple candidate outputs of their baseline system. They adopt the averaged perceptron (Collins, 2002) as their learning framework for the reranking system, which is integrated with several additional features. Their final system impressively increases the ACC to 0.477, while the Mean F-score is 0.740. Their work indicates the importance of reranking in a machine transliteration system.

Meanwhile we should notice we use a different testing set from those two submitted systems, so the results are not completely comparable. However, it still indicates that the performance of our transliteration system is at least close to the state-of-art systems.

5 Experiments on Different Language Sets

5.1 Language Specific Systems

The most common source language in transliteration tasks is English. However, there are also many source names that have other language origins. If we use a transliteration model that is trained on the English data set to process non-English names, we may get errors as those languages normally do not share the same pronunciation rules with English. Considering that name transliteration is phonetic-based, we therefore train different transliteration models with different language sets to build language specific transliteration systems.

As introduced previously, in this thesis, we obtained 14 different language sets from the Translation Dictionary for Foreign Names. We build 14 different transliteration systems respectively for those language sets. For the data of each language set, we randomly select 1/10 of the terms to form the testing set, and we use the rest as the training set. The training set is then applied to both the baseline and modified systems that are constructed in the previous section to do further experiments.

In some languages such as Spanish, French and Portuguese, there are some vowels that have accented marks, for examples, "á", "é" and "ó". Before the data sets are applied to our transliteration systems, including the generic system introduced in the next section, we replace those vowels with their ordinary base forms. Similarly to tone information, we also assume that accents are irrelevant to transliteration and the accented vowels are essentially the same as the non-accented ones; therefore, they should not be processed as different letters.

5.2 Generic System

Apart from the language specific systems, we also train a generic system without language source discrimination. We assemble all the training sets of different languages and merge them together into a larger training set for our generic system. It should be noted that there are some names with identical Chinese transliterations that appear in different sub language sets. In the merged training set we only count them as one training instance. However, it is also very common that some terms have the same source name but different transliterations in different language sets. In this case we consider those terms as different training instances and keep them all in the merged training set. The final training set for our generic system contains 471,354 different terms.

Similar to the language specific models, we use this merged training set to train both our baseline and modified systems. The two systems are then tested with all the testing sets of different language sets. We compare the performances with the language specific systems afterwards.

5.3 Results and Evaluations

Table 5.1 and 5.2 show the performances of our language specific systems as well as generic system on different language sets. MFS is short for Mean F-score.

Generally, as expected, we see that the language specific systems outperform the generic system. For most language sets, the language specific systems have significantly higher ACCs as well as Mean F-scores, which indicates that the language specific systems are very effective on most language sets. Especially for German, Spanish, Turkish and Russian, the ACCs and Mean F-scores achieved by the generic system are rather low while the language specific systems greatly boost the performance, which makes their ACCs all above 0.5 and Mean F-scores over 0.8.

For many other language sets in particular as Czech, Finnish, Italian, Romanian, Serbian and Swedish, the generic system yields already rather high ACCs and Mean F-scores. However, the language specific systems still lead to quite impressive improvements. The ACCs achieved by the language specific systems are all around 0.6 and the Mean F-scores are all close to 0.85, which is rather impressive for a machine transliteration system.

For the remaining language sets, which are English, French, Hungarian and Portuguese, the generic system performs very poorly while the results of the language specific systems are not so satisfying either. Nevertheless, we should be aware that there are still huge discrepancies between the performances of the two systems. The language specific systems still prove to be advantageous.

Overall, we can conclude that the language specific systems are better than the generic system in terms of the transliteration precision. This reveals the importance of discrimination of language origins in machine transliteration.

Moreover, we can see that our modified system performs better than the baseline system in both frameworks, the language specific systems as well as the generic system, on most language sets. The modified system generally achieves larger improvement when the ACC and Mean F-score of the baseline system are low. In this case, the modification seem to be more valuable in the generic system.

In the generic system, the modified system is particularly effective on some language sets, such as Czech, German, Romanian and Turkish. French is the only set that the modified system has detrimental effects.

On the other hand, the improvements achieved by the system modification in the language specific systems are not so impressive on most language sets, while for English and German, the ACCs of the modified systems are still around 10% higher. We can see that on the Turkish set as well as the Portuguese set, the modified systems perform worse than the baseline systems. However, on the Turkish set in the generic system, the ACC of the modified system is 22% higher than the baseline system, which is significantly better.

Languages	Language Specific Systems				
	Size of training set	Baseline		Modified	
		ACC	MFS	ACC	MFS
Czech	32,189	0.547	0.849	0.582	0.864
English	45,239	0.331	0.688	0.364	0.710
Finnish	16,458	0.647	0.876	0.663	0.878
French	75,568	0.422*	0.770	0.424*	0.768
German	46,118	0.507	0.825	0.573	0.854
Hungarian	25,600	0.398	0.747	0.426	0.760
Italian	55,057	0.598	0.859	0.623	0.887
Portuguese	9,641	0.465*	0.784	0.454*	0.773
Romanian	26,950	0.548	0.833	0.560	0.837
Russian	45,249	0.512	0.833	0.530	0.837
Serbian	31,548	0.585	0.860	0.591	0.861
Spanish	27,600	0.579	0.856	0.586	0.858
Swedish	27,674	0.616*	0.867	0.625*	0.870
Turkish	13,609	0.650	0.865	0.632	0.865

Table 5.1: Performances of the language specific systems on different language sets

Languages	Generic System			
	Baseline		Modified	
	ACC	MFS	ACC	MFS
Czech	0.345	0.746	0.423	0.798
English	0.204*	0.585	0.211*	0.605
Finnish	0.467	0.801	0.507	0.819
French	0.233	0.619	0.197	0.607
German	0.269	0.675	0.325	0.719
Hungarian	0.203	0.584	0.235	0.618
Italian	0.441	0.779	0.478	0.797
Portuguese	0.346*	0.695	0.361*	0.709
Romanian	0.413	0.776	0.483	0.807
Russian	0.329	0.751	0.357	0.765
Serbian	0.440	0.794	0.479	0.810
Spanish	0.255	0.661	0.274	0.682
Swedish	0.382	0.749	0.428	0.782
Turkish	0.324	0.720	0.396	0.754

Table 5.2: Performances of the generic system on different language sets

In order to further investigate the significance of differences between the precisions of different machine transliteration systems, we perform the Wilcoxon signed-rank test (Wilcoxon, 1945) on the ACC scores that are evaluated on our testing data. For each language set, first we test the significance of difference between the language specific systems and the generic system respectively using the baseline system and the modified system. We find that the differences between the language specific systems and generic system are all highly

statistically significant with p-values far below any critical value according to the test results, which again indicates the importance of specifying language origins in machine transliteration.

Next, we perform the signed-rank test to see the significance of the difference between the baseline system and the modified system under both the language specific systems as well as the generic system. The majority of the differences between the tested pairs are significant with p-values below 0.01. For Serbian under the language specific system, the calculated p-value is above 0.01 but less than 0.05. There are a few tested pairs whose calculated p-values are above 0.05 and therefore we cannot reject the H_0. We mark those pairs with "*" in Table 5.1 and 5.2. We see that for Portuguese, the differences between the modified system and baseline system are not significant under either the language specific system or the generic system. The modified system is not significantly worse under the language specific system while it is not significantly better under the generic system either. However, according to the result of signed-ranked test, we do see that the modified system is significantly detrimental for French under the generic system. But overall the results show that our modifications to the baseline system are still effective.

5.4 Analysis of Language Specific Transliteration

In these experiments, we apply the same transliteration model to different language sets and observe that it performs very differently. It is obvious that some languages are more difficult for the transliteration model to process due to their essential characteristics of pronunciation.

From the results table we see that our system achieves high ACCs and Mean F-scores on Finnish, German, Italian, Spanish and Swedish. Those languages have comparatively explicit pronunciation rules. The pronunciation of the letters is less flexible in different contexts.

On the other hand, the system gets low evaluations on English, French and Hungarian. As the most international language, English assimilates different names from the other origins which makes the pronunciation much less consistent than in other languages. Meanwhile, unpronounced letters are fairly common in French. Hungarian has 14 vowel phonemes and 25 consonant phonemes as well as their different variations. Those all bring difficulties to the transliteration system as we can observe in the results of our experiments.

Moreover, it is also possible that some of the languages that gain higher evaluation scores contain simpler source names that are easier to transliterate. Thus, we use two features to further investigate the testing data. First we calculate the average length of all the source names in the testing set. We assume that the longer source names are the more complex so that they are more difficult for the system to transliterate accurately. We also compute how many letters are on average transliterated by one Chinese character, which is simply the ratio of the numbers of the source and target characters. More letters in the source languages transliterated by fewer Chinese characters means that either there are more silent letters in the source languages or the transliterated substrings are more complicated, therefore the source names are more likely to

be transliterated incorrectly. This feature is definitely related to the essential phonetic characteristics of the languages. The statistics are shown in Table 5.3.

Languages	Average Length of the Source Names	Ratio of the Length of Source Characters and Target Characters
Czech	7.50	1.81
English	6.98	2.16
Finnish	7.51	2.07
French	7.24	2.32
German	7.65	2.13
Hungarian	6.90	1.94
Italian	7.79	2.09
Portuguese	7.04	1.97
Romanian	7.24	1.84
Russian	8.36	1.91
Serbian	7.19	1.76
Spanish	6.94	1.90
Swedish	7.51	1.99
Turkish	6.66	1.73

Table 5.3: Further investigation of the testing data sets

The source names in Russian have the largest average length. Considering that Russian data are transcribed into Latin letters from the original Cyrillic alphabet, we regard it as an exception. Apart from Turkish, there is no significant difference between all the other language sets in average length. As an Altaic language, Turkish is a bit different from the other European languages. Both of the measures for Turkish are the smallest. Meanwhile we can see that for French, more letters are transliterated by a single Chinese character on average, which illustrates that the French pronunciation system makes machine transliteration more difficult.

For most of the language sets, the size of the training data seems to be big enough for the transliteration systems to achieve good precision. For some sets, it is likely to get higher evaluations if we would use a larger training set, for instance Portuguese. Considering Portuguese is phonetically similar to Spanish and Italian, the transliteration system for Portuguese should be able to perform equally well as the Spanish and Italian systems.

For Turkish, the training data are not sufficient either. For example, there is an instance "Subuğ" in the testing set that is not processed by the transliteration system. "ğ" never appeared as an independent substring in the training set. It becomes an unknown fragment that cannot be handled by the decoding algorithm, which fails the entire transliteration process. The letters that are used in a certain language are very limited. If we have enough training data, the problem should be avoided. However, our transliteration system for Turkish still manages to get considerably high overall evaluations.

5.5 Further Discussion

As stated in the previous section, it is clear that the language specific systems perform better than the generic system in terms of precision. Moreover, it is important to notice that the language specific systems are much more efficient. The efficiency of the transliteration system mainly depends on the size of the training set. In order to build a generic transliteration system which covers many different languages, normally we have to use a large training set, while the training sets used for the language specific systems are much smaller and therefore the search space for the decoding algorithm is also smaller. Take the transliteration of Finnish as an example, in our experiments, it takes 2.95 seconds for the language specific system to finish transliterating the complete test set while the generic system requires 146.4 seconds: an enormous difference.

However, the language specific systems also have limitations. As they are oriented to certain languages, sometimes it is difficult to determine which model should be applied on the source names if the language origin is not explicit. Certainly we can judge the language origin from the context but it is not always reliable. In fact it is fairly common that the source name from one language appears in the context of another language. For example,

"Julio Iglesias tävlade för Spanien med låten Gwendolyne i Eurovision Song Contest 1970 och slutade på fjärde plats."

"Julio Iglesias" is a Spanish name. Even though it appears in a Swedish text, we should still use the language specific system built for Spanish to perform the transliteration.

An alternative option to solve the problem is to use a classifier to determine the source name's language origin. However, as there are so many different languages and the names are normally very short, it is very difficult to have a reliable language identifier.

Furthermore, we cannot always retrieve sufficient data to train language specific systems for all different languages. The Transliteration Dictionary for Foreign Names that we use in this thesis contains source names from more than 100 different language origins, while a number of them only have a few instances. In this case, we can either use the generic system or a language specific system trained on a related language.

Swedish, Danish and Norwegian are all Nordic Germanic languages, derived from Old Norse. We have sufficient data to train a Swedish transliteration system, whereas the data for Danish and Norwegian are not enough. The dictionary contains 851 Danish instances and 2,189 Norwegian instances. We apply our language specific system for Swedish on the Danish and Norwegian sets, and compare the performance with the generic system. In order to make the decoder for Swedish capable of fully processing the Danish and Norwegian data, we replace the letters "ø" and "æ" in Danish and Norwegian with the Swedish "ö" and "ä" in the source names before transliterating. The detailed results are in Table 5.4 and 5.5.

We can see that generally neither of the two language sets achieves good evaluations. They get very low ACCs as well as Mean F-scores within the generic system. However, the language specific system for Swedish performs

Languages	Size of testing set	Swedish Language Specific System			
		Baseline		Modified	
		ACC	MFS	ACC	MFS
Danish	851	0.286	0.684	0.294	0.696
Norwegian	2,189	0.303	0.717	0.301	0.718

Table 5.4: Performances of the language specific system for Swedish on Danish and Norwegian

Languages	Size of testing set	Generic System			
		Baseline		Modified	
		ACC	MFS	ACC	MFS
Danish	851	0.208	0.631	0.228	0.664
Norwegian	2,189	0.222	0.652	0.255	0.687

Table 5.5: Performances of the generic system on Danish and Norwegian

significantly better, which at least indicates that the language specific system of a related language is very effective and can be superior to the generic system in machine transliteration.

6 Conclusion

6.1 Summary

Machine transliteration is an effective approach to complement the overall performance of the machine translation system. It also plays an important role in cross-language information retrieval. Chinese as a major Asian language is often the target language in many machine transliteration tasks. Due to the essential characteristics of the Chinese language, it is challenging to build an effective transliteration model which is universal and efficient as well as capable of achieve good precisions in transliterating named entities from different languages into Chinese.

In this thesis, focusing on the transliteration of names, we construct a HMM based machine transliteration system. We modify the classical HMM model by adding pinyin as an intermediate phonetic representation in the system. The M2M Aligner is used as the critical tool to segment and align the training data via unsupervised learning. We implement the Viterbi Algorithm in the decoder to guarantee the efficiency of the system. Our baseline obtains ACC of 0.336 and Mean F-score of 0.691 on the testing data from the NEWS 2010 workshop. Additionally, we use four approaches to modify the baseline system: pre-contracting letter combinations for M2M Aligner, trimming low frequent terms, adding penalty scores as well as preprocessing the letter "x". The integrated modified system achieves significant improvements. The ACC is increased to 0.370 and Mean F-score reaches 0.714. The precision of the modified transliteration system is comparable to state-of-art machine transliteration systems that are submitted in NEWS 2010.

Both our baseline system and modified system are applied on 14 different language sets. Using the different training data, we build distinct language specific systems which are directed towards certain languages. Meanwhile we train a generic system using the assembled training data to compare with the language specific systems. Generally the language specific systems outperform the generic systems in precision. The language specific systems are particular effective on some languages such as German, Spanish, Italian and Russian. Compared to the baseline transliteration system, our modified system is beneficial in most experiments. We also notice that some languages are more difficult for the transliteration to process, for example English, French and Hungarian due to their essential characteristics. Furthermore, the language specific systems are also more efficient because of the smaller search space for the decoder.

On the other hand, even though the language specific systems show considerable superiorities, there are also some disadvantages. The language specific systems are limited to certain language sets, and therefore sometimes a highly

accurate classifier is required to determine the language origin first in practical transliteration problems. The experiment on Danish and Norwegian using the Swedish transliteration system indicates that it is a good alternative to apply the language specific system of a related language to the languages that do not have sufficient training data.

Overall, the experiment results on our machine transliteration models have reached our initial expectation. We have successfully built the transliteration system and made it applicable to different languages with satisfying levels of precision.

6.2 Future Work

Even though our transliteration model gets high evaluation scores, there is still potential to improve its performance. As stated in Chen et al. (2010)'s research, reranking the different candidate outputs is a very effective approach to further increase the precision of the transliteration system. Currently our system only returns one optimal output; we will modify our system in the future to make it produce several candidates so that we can apply reranking. On the other hand, we should notice that adding reranking as a post procedure also makes the entire transliteration system less efficient.

Additionally, we can build transliteration systems which are oriented to transliteration of the other Asian languages, such as Japanese, Korean and Vietnamese. This should be comparatively less difficult as their alphabets all have strong connections with Chinese characters. Our current transliteration system is not capable of processing Asian source names while in reality transliterations between these languages are fairly common.

We will also make an effort to collect further data sets that can be used as the training data to establish more language specific systems. Meanwhile, we are interested in categorizing the languages into several general classes based on their relations and building some more universal as well as comparatively accurate transliteration systems. For example, we use Spanish, Italian, Portuguese and Romanian to train a system that can also be used for the transliteration of some minor Romance languages, such as Haitian, Sicilian and Catalan. If we have several bigger language classes, it will also be more practical to train a classifier to determine which class the source name belongs to. Directly identifying the explicit language origin of the source names is much more difficult.

Moreover, we would like to publish our implementation on the internet as an application so people can use it if they are interested in knowing how their names are transliterated in Chinese. We are also planning to test our transliteration system within a machine translation framework to further investigate its performance.

Bibliography

Abduljaleel, N. and Larkey, L. (2003). Statistical transliteration for English-Arabic cross language information retrieval. In Proceedings of the 20th international conference on Information and Knowledge Management, page 139–146, LA, USA.

Aramaki, E. and Abekawa, T. (2009). Fast decoding and easy implementation: Transliteration as sequential labeling. In NEWS '09 Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration, Stroudsburg, PA, USA.

Arbabi, M., Fischtal, S., Cheng, V., and Bart, E. (1994). Algorithms for Arabic name transliteration. IBM Journal of Research and Development, 38(2):183–193.

Chen, Y., Ouyang, Y., Li, W., Zheng, D., and Zhao, T. (2010). Using deep belief nets for chinese named entity categorization. In Proceedings of the 2010 Named Entities Workshop, pages 102–109, Uppsala, Sweden. Association for Computational Linguistics.

Chen, Y., Wang, R., and Zhang, Y. (2011). Statistical machine transliteration with multi-to-multi joint source channel model. In Proceedings of the Named Entities Workshop Shared Task on Machine Transliteration, Chiang Mai, Thailand.

Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In In Proceedings of EMNLP-2002, page 1–8.

Forney, G. J. (1973). The Viterbi algorithm. Proceedings of the IEEE, 61:268 – 278.

Ghahramani, Z. (2001). An introduction to Hidden Markov Models and Bayesian networks. International Journal of Pattern Recognition and Artificial Intelligence, 15(1):9–42.

Jiampojamarn, S., Dwyer, K., Bergsma, S., Bhargava, A., Dou, Q., Kim, M.-Y., and Kondrak, G. (2010). Transliteration generation and mining with limited training resources. In Proceedings of the 2010 Named Entities Workshop, pages 39–47, Uppsala, Sweden. Association for Computational Linguistics.

Jiampojamarn, S., Kondrak, G., and Sherif, T. (2007). Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion. In

Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference, pages 372–379, Rochester, New York. Association for Computational Linguistics.

Jiang, X., Sun, L., and Zhang, D. (2009). A syllable-based name transliteration system. In NEWS '09 Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration, Stroudsburg, PA, USA.

Jung, S., Hong, S., and Paek, E. (2000). An English to Korean transliteration model of extended markov window. In Proceedings of the 18th conference on Computational Linguistics, page 383–389, Saarbrucken, Germany.

Jurafsky, D. and Martin, J. H. (2008). Speech and Language Processing. Prentice Hall, New Jersey, US, 2 edition.

Karimi, S., Scholer, F., and Turpin, A. (2011). Machine transliteration survey. ACM Computing Surveys (CSUR), 43.

Knight, K. and Graehl, J. (1997). Machine transliteration. In Proceedings of the 35th Annual meeting of the ACL and Eighth Conference of the European chapter of the ACL, page 128–135, Madrid, Spain.

Koehn, P., Och, F., Franz, J., and Marcu, D. (2003). Statistical phrase-based translation. In Proceedings of the 2003 conference on the North American Chapter of the Association for Computational Linguistics on Human Language Technology, page 48–54.

Kwong, O. Y. (2009). Phonological context approximation and homophone treatment for news 2009 English-Chinese transliteration shared task. In NEWS '09 Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration, pages 76–79, Stroudsburg, PA, USA.

Li, H., Kumaranz, A., Zhang, M., and Pervouchine, V. (2010a). Report of NEWS 2010 transliteration generation shared task. In NEWS '10 Proceedings of the 2010 Named Entities Workshop: Shared Task on Transliteration, Uppsala, Sweden.

Li, H., Kumaranz, A., Zhang, M., and Pervouchine, V. (2010b). Whitepaper of NEWS 2010 shared task on transliteration generation. In NEWS '10 Proceedings of the 2010 Named Entities Workshop: Shared Task on Transliteration, Uppsala, Sweden.

Li, H., Zhang, M., and Su, J. (2004). A joint source-channel model for machine transliteration. In Proceedings of the 42nd meeting of the Association for Computational Linguistics, page 159–166, Barcelona, Spain.

Nabende, P. (2009). Transliteration system using pair HMM with weighted fsts. In NEWS '09 Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration, Stroudsburg, PA, USA.

Nabende, P. (2011). Applying Dynamic Bayesian Networks in Transliteration Detection and Generation. PhD thesis, University of Groningen, Zutphen, The Netherlands. Wörhmann Print Service.

Oh, J. H., Choi, K. S., and Isahara, H. (2006). A comparison of different machine transliteration models. Journal of Artificial Intelligence Research, 27:119–151.

Ravi, S. and Knight, K. (2009). Learning phoneme mappings for transliteration without parallel data. In NAACL '09 Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pages 37–45, Stroudsburg, PA, USA.

Ristad, E. S. and Yianilos, P. N. (1998). Learning string edit distance. IEEE Transactions on Pattern Recognition and Machine Intelligence, 20(5):522–532.

Roger, M. (1992). A description of a computer-usable dictionary file based on the Oxford advanced learner's dictionary of current English. Oxford Text Archive.

Tiedemann, J. and Nabende, P. (2009). Translating transliterations. International Journal of Computing and ICT Research, pages 33–41.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. Biometrics Bulletin, 1(6):80–83.

Xia, D. (1993). Translation Dictionary for Foreign Names. China Translation and Publishing Corporation, Beijing, China.

Zhang, M., Duan, X., Pervouchine, V., and Li, H. (2010). Machine transliteration: leveraging on third languages. In COLING '10 Proceedings of the 23rd International Conference on Computational Linguistics: Posters, pages 1444–1452, Stroudsburg, PA, USA.

Zhou, Y. (2009). Maximum n-gram hmm-based name transliteration: Experiment in NEWS 2009 on English-Chinese corpus. In NEWS '09 Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration, pages 128–131, Stroudsburg, PA, USA.