Machine Transliteration of Proper Names

David Matthews

Master of Science

School of Informatics

University of Edinburgh

2007

Abstract

Transliteration (or *forward transliteration*) is the process of mapping source language phonemes or graphemes into target language approximations, the reverse process is called *back transliteration*. Most previous approaches to the problem have sought to model transliteration as a combination of both grapheme and phoneme level transformations (Knight and Graehl, 1997; Stalls and Knight, 1998; Al-Onaizan and Knight, 2002).

This thesis describes and evaluates an automatic transliteration system built using Moses (Koehn et al., 2007), a Phrase-Based Statistical Machine Translation system, with which we model transliteration as translation only at surface level. Separate translation and target side language models are constructed and combined during decoding to find the most likely transliteration.

Moses outperforms our baseline model in both directions in both English-Chinese and Arabic-English. The best forward transliteration accuracy achieved from English to Chinese was 37.8%, the best back transliteration accuracy from Chinese to English was 34.8%. The best forward transliteration accuracy achieved from Arabic to English was 43.0%, the back transliteration accuracy from English to Arabic was 39.2%. Moses' performance is comparable to recent work in both English-Chinese (Jiang et al., 2007) and Arabic-English (Zhao et al., 2007).

Acknowledgements

I would like to thank Miles Osbourne for his advice and enthusiasm throughout the supervision of this project, the long line of people involved in obtaining the Arabic-English data, Abby Levenberg for nothing in particular and last but not least my mother for her support and encouragement throughout the year.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*David Matthews*)

Contents

1 Introduction **1**

2 Background **5**

2.1 Noisy-Channel Transliteration 5

2.2 Extensions to Noisy-Channel Transliteration 6

2.3 Alternative Approaches 7

2.4 Phrase-Based Statistical Machine Translation 8

2.4.1 Language Models 10

2.4.2 Optimisation & Evaluation 11

3 Approach **13**

3.1 Baseline 13

3.2 Transliterating with Moses 13

4 Experimental Framework **17**

4.1 Data 17

4.1.1 Chinese 17

4.1.2 Arabic 19

4.2 Software 21

4.2.1 Moses 21

4.2.2 GIZA++ 22

4.2.3 SRILM 22

4.2.4 Mert 22

4.2.5 Bleu 22

4.2.6 Python 23

4.3 Metrics 23

4.4 Experiments 24

4.4.1 Baseline 24

4.4.2 Default Settings 24
4.4.3 Transliteration Model 25
4.4.4 Language Model 25
4.4.5 Optimising with Mert 26

5 Results **29**
5.1 Chinese Results 29
5.1.1 Baseline & Default Settings 29
5.1.2 Transliteration Model 32
5.1.3 Language Model 32
5.1.4 Optimising with Mert 35
5.1.5 Example Transliterations 37
5.2 Arabic Results 38
5.2.1 Baseline & Default Settings 38
5.2.2 Transliteration Model 38
5.2.3 Language Model 41
5.2.4 Optimising with Mert 41
5.2.5 Example Transliterations 44

6 Conclusion **47**
6.1 Summary 47
6.2 Future Work 48

Bibliography **49**

List of Figures

1.1 Examples of Japanese-English Forward Transliteration 1
1.2 Examples of Russian-English Back Transliteration 1

2.1 Example Phrase Alignment 9

3.1 Example Transliteration Options 14

4.1 Sample English-Chinese Data 18
4.2 Sample English-Chinese Preprocessed Data 18
4.3 Sample Arabic-English Data 20
4.4 Sample Arabic-English Preprocessed Data 20

5.1 English-Chinese Maximum Phrase Length (Forward) 30
5.2 English-Chinese Maximum Phrase Length (Back) 30
5.3 English-Chinese Transliteration Model Training Data Size (Forward) . 31
5.4 English-Chinese Transliteration Model Training Data Size (Back) . . 31
5.5 English-Chinese Language Model N-Gram Order (Forward) 33
5.6 English-Chinese Language Model N-Gram Order (Back) 33
5.7 English-Chinese Language Model Training Data Size (Forward) . . . 34
5.8 English-Chinese Language Model Training Data Size (Back) 34
5.9 Arabic-English Maximum Phrase Length (Forward) 39
5.10 Arabic-English Maximum Phrase Length (Back) 39
5.11 Arabic-English Transliteration Model Training Data Size (Forward) . 40
5.12 Arabic-English Transliteration Model Training Data Size (Back) . . . 40
5.13 Arabic-English Language Model N-Gram Order (Forward) 42
5.14 Arabic-English Language Model N-Gram Order (Back) 42
5.15 Arabic-English Language Model Training Data Size (Forward) 43
5.16 Arabic-English Language Model Training Data Size (Back) 43

List of Tables

5.1 English-Chinese Baseline & Default Settings 29
5.2 English-Chinese Forward Transliteration Final System 36
5.3 English-Chinese Back Transliteration Final System 36
5.4 English-Chinese Back Transliteration Gigaword Language Model . . 36
5.5 Example English-Chinese Forward Transliterations 37
5.6 Example English-Chinese Back Transliterations 38
5.7 Arabic-English Baseline & Default Settings 38
5.8 Arabic-English Forward Transliteration Final System 45
5.9 Arabic-English Back Transliteration Final System 45
5.10 Arabic-English Forward Transliteration Gigaword Language Model . 45
5.11 Example Arabic-English Forward Transliterations 46
5.12 Example Arabic-English Back Transliterations 46

Chapter 1

Introduction

Transliteration (or *forward transliteration*) is the process of transforming source language phonemes or graphemes into target language approximations. The reverse of this process, i.e. transforming target language approximations back into their original source language is called *back transliteration*. As forward transliteration is a lossy process, due to the lack of direct correspondence between languages' phonetic systems, back transliteration is far from trivial.

McDonald's → マクドナルド (ma-ku-do-na-ru-do)
カラオケ (ka-ra-o-ke) → karaoke

Figure 1.1: Examples of Japanese-English Forward Transliteration

Interest in automatic proper name transliteration has grown in recent years due to its ability to help combat *transliteration fraud* (The Economist Technology Quarterly, 2007), the process of slowly changing a transliteration of a name to avoid being traced by law enforcement and intelligence agencies.

Марков ← Markov
spam, spammer ← спам, спамеров

Figure 1.2: Examples of Russian-English Back Transliteration

The ability to transliterate proper names also has applications in Statistical Machine Translation (SMT). SMT systems are trained using large parallel corpora, while these corpora can consist of several million words they can never hope to have complete coverage especially over highly productive word classes like proper names. When translating a new sentence SMT systems draw on the knowledge acquired from their

training corpora, if they come across a word not seen during training then they will at best either drop the unknown word or copy it into the translation and at worst fail.

Current methods of automatically evaluating SMT systems rely on counting exact matches of sequences of words of differing lengths, e.g. Bleu (Papineni et al., 2001), therefore only exact transliterations of unknown names will give an increase in performance. Much like translation there are typically many acceptable answers, for example English transliterations of محمد (Muhammad) include Mohamad, Mohamed, Mohammad, Mohammed, Muhamad, Muhamed, Muhammad, Muhammed. In order to improve performance within an SMT system the *intended* transliteration rather than an *acceptable* transliteration needs to found.

To widen the pool of acceptable translations and hence transliterations multiple references are sometimes provided but even with these improving SMT performance via transliterating unknown names is still a difficult task.

Cross Language Information Retrieval (CLIR) can also benefit from transliterations of unknown words and proper names (AbdulJaleel and Larkey, 2003; Virga and Khudanpur, 2003). By its nature CLIR applications can see increases in recall if not in precision when utilising imperfect transliterations.

In this thesis we give results of using Moses (Koehn et al., 2007), a Phrase-Based Statistical Machine Translation, to conduct experiments transliterating proper names between both English-Chinese and Arabic-English. Separate transliteration and target side language models are constructed and combined during decoding to find the most likely transliteration. Various experiments were performed to find optimal parameter settings and investigate the effects of varying the size of both language and transliteration models.

Moses outperforms our baseline model in both directions in both language pairs. The best forward transliteration accuracy achieved from English to Chinese was 37.8%, the best back transliteration accuracy from Chinese to English was 34.8%. The best forward transliteration accuracy achieved from Arabic to English was 43.0%, the back transliteration accuracy from English to Arabic was 39.2%.

The rest of this thesis is organised as follows:

- Chapter 2 reviews previous work in Machine Transliteration and introduces the Phrase-Based Statistical Machine Translation framework of which Moses is an implementation.

- Chapter 3 describes our baseline and our approach to transliteration using Moses.
- Chapter 4 details the data, software and metrics used as well as giving descriptions of our experiments.
- Chapter 5 gives the results of the experiments.
- Chapter 6 offers conclusions and points to areas of future work.

Chapter 2

Background

The need for automatic transliteration in both its major areas of application, SMT and CLIR, stems from out-of-vocabulary words (OOVs), i.e. words for which a system has no translation. OOVs tend to be proper names due to the size and high productivity of the class.

Translations between English and languages that can be viewed as having a superset of the English alphabet, e.g. most European languages, rarely, if ever transliterate names, for example Guillaume (French) and Jorge (Spanish) are usually left unchanged rather than translated into their Anglicised equivalents William and George.

Transliteration becomes necessary when two languages' character sets do not have any characters in common as is the case with Chinese, Japanese, Korean and Arabic when paired with English. It is these pairings where the majority of previous research into machine transliteration has been focused.

2.1 Noisy-Channel Transliteration

Whilst rule-based systems are useful up to a point they rely on strict adherence to a transliteration standard laid down by a government or other official body, transliterations that differ from a standard have to be handle via exceptions. Manual creation of such exceptions is time consuming and can struggle to achieve the robustness of automatic statistical methods.

The first major piece of work on Statistical Machine Transliteration was Knight and Graehl (1997). Back transliteration of Japanese into English is modelled generatively; the problem is broken down into a number of steps and recombined using Bayes' rule:

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, the syllabary that is used to write transliterations of foreign words
5. The Japanese is written

These subproblems are then cast as probabilities:

1. $p(w)$ - generates written English word sequences
2. $p(e|w)$ - pronounces English word sequences
3. $p(j|e)$ - converts English sounds into Japanese sounds
4. $p(k|j)$ - converts Japanese sounds into Japanese katakana
5. $p(o|k)$ - introduces misspellings caused by Optical Character Recognition (OCR)

Now given a string of Japanese katakana o taken from an OCR system the system finds the English word sequence $\hat{w}$ that maximises the sum over all e, j and k, of:

$$p(w)p(e|w)p(j|e)p(k|j)p(o|k) \tag{2.1}$$

The generation of English word sequences $p(w)$ is implemented as a weighted finite-state acceptor (WFSA) and the rest of the distributions as weighted finite-state transducers (WFSTs), each trained on an appropriate corpus. The most probable English transliteration is computed using Dijkstra's shortest-path algorithm (Dijkstra, 1959).

2.2 Extensions to Noisy-Channel Transliteration

Since Knight & Graehl's work there has been an ongoing effort to adapt, extend and refine the techniques used. Stalls and Knight (1998) adapt the noisy-channel transliteration model given above for Arabic. The first two components of the model $p(w)$ and $p(e|w)$ are reused and combined with a component to model the direct transliteration of English phonemes to Arabic writing $p(a|e)$ trained over a small hand crafted bilingual dictionary. The fact that English vowels could produce certain Arabic letters only in certain contexts is used to markup the vowels with positional information in order to learn these rules during training.

Al-Onaizan and Knight (2002) seek to overcome the limitation of Stalls and Knight (1998) to only handle English named entities (NEs) with known pronunciations and the fact that many transliterations follow how a word is spelt rather than how it sounds. They propose a spelling-based model $p_s(w|a)$ which models the direct translation of surface forms and linearly combine it with the phonetic-based model $p_p(w|a)$:

$$p(w|a) = \lambda p_s(w|a) + (1-\lambda) p_p(w|a) \tag{2.2}$$

In addition to the new spelling-based model the Arabic name to be transliterated is preprocessed to correct potential spelling mistakes and transliteration candidates generated via the spelling-based model are post-processed to eliminate any with zero web counts.

Al-Onaizan and Knight (2001) take this post-processing further. They assume a story being reported in a source language document has been reported in the chosen target language, NEs identified in a similar target language document can be used to rescore a ranked list of translation candidates generated from source NEs using the methods laid out in Stalls and Knight (1998) and Al-Onaizan and Knight (2002).

To find a document in the target language that is similar enough to the original source document to be useful a web search query is formulated from translations of keywords found in the original document. Keywords extracted from results of the search are used to expand the original query in order to hone in on a useful document.

A number of other rescoring methods are employed including web counts of the translation candidate in isolation, contextual web counts, i.e. the candidate word plus a translated keyword combined with a Boolean AND, and rescoring abbreviated entities using counts of the unabbreviated form, e.g. using counts of "the House of Representatives" to rescore "the House". All these methods show some improvement over the basic noisy-channel model.

The model handles locations and organisations differently to personal names as the former are often a combination of translations and transliterations whereas the later are purely transliterated in the vast majority of cases. All permutations of translations and transliterations are generated for both locations and organisations and scored against a large news corpus with a modified IBM Model 1 probability (Brown et al., 1993).

2.3 Alternative Approaches

AbdulJaleel and Larkey (2003) break training down into two stages, first learning use-

ful segmentations of the source language and then learning a transliteration model over these segments. Kang and Choi (2000) model both transliteration and back transliteration of English-Korean via decision tree learning. Chen et al. (2003) counter the problem that some NEs are translated as a mixture of transliteration and translation with a frequency-based approach, starting with the assumption that transliterated terms will appear far less frequently in a parallel corpus than translated terms. See Oh et al. (2006) for further comparison of grapheme-based, phoneme-based and hybrid systems.

2.4 Phrase-Based Statistical Machine Translation

Phrase-Based SMT has its roots in the work of Brown et al. (1993) who developed a word-based noisy-channel model of translation much like that used by Knight and Graehl (1997) to model back transliteration as described in Section 2.1.

Given a foreign sentence[1] f the model provides us with the probability $p(e|f)$ of an English sentence e. Bayes' rule is applied to allow us separately model the translation probability $p(f|e)$, which ensures that the English produced is an appropriate translation of the foreign sentence, and that of the English sentence $p(e)$, which ensures fluent English output:

$$p(e|f) = \frac{p(e)p(f|e)}{p(f)} \tag{2.3}$$

The probability of the foreign sentence $p(f)$ can be dropped as it is constant and would not have any effect on finding the English sentence $\hat{e}$, which maximises the equation $p(e)p(f|e)$:

$$\hat{e} = argmax_e\, p(e)p(f|e) \tag{2.4}$$

Word-Based models give us a one-to-many mapping between words in parallel corpora, this is not ideal as there are many instances of a many-to-one or a many-to-many mapping giving a natural translation or transliteration, see Figure 2.4. The Phrase-Based model put forward by Koehn et al. (2003) was developed in part to deal with this deficiency of Word-Based models. The starting point is the same as the Word-Based model as given in Equation 2.4, from there things begin to differ, we decompose $p(f|e)$ to:

[1] This thesis sticks to the tradition within the SMT community to always translate from a foreign language into English when giving examples and as we are modelling transliteration as translation we substitute character for word and name for sentence

$$p(\bar{f}_1^I|\bar{e}_1^I) = \prod_{i=1}^{I} \phi(\bar{f}_i|\bar{e}_i) d(start_i - end_{i-1} - 1) \tag{2.5}$$

The translation starts with the foreign sentence f being broken down into I phrases, $f_1 \ldots f_I$, each segmentation being equally likely. Now each foreign phrase f_i is translated to an English phrase e_i.

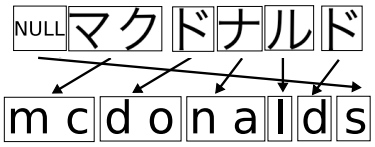

Figure 2.1: Example Phrase Alignment

An additional model, $d(start_i - end_{i-1} - 1)$, referred to as the distortion model, handles reordering of phrases. As we are dealing with transliteration we assume that all translations will be monotone, i.e. no reordering, and so can ignore this. Bringing all the components together we get:

$$\hat{e} = argmax_e \prod_{i=1}^{I} \phi(\bar{f}_i|\bar{e}_i) \prod_{i=1}^{|e|} p_{LM}(e_i|e_1 \ldots e_{i-1}) \tag{2.6}$$

We can generalised this setup in a *log-linear* model (Och and Ney, 2001) that allows us to model any number of different aspects of the data as features, these are combined with weights giving more or less significance to each component:

$$\hat{e} = argmax_e \prod_{i=1}^{I} \phi(\bar{f}_i|\bar{e}_i)^{\lambda_\phi} \prod_{i=1}^{|e|} p_{LM}(e_i|e_1 \ldots e_{i-1})^{\lambda_{LM}} \tag{2.7}$$

Equation 2.7 can be reformulated as:

$$\begin{aligned} \hat{e} = argmax_e \, exp \, (\lambda_\phi \sum_{i=1}^{I} log \, \phi(\bar{f}_i|\bar{e}_i) + \\ \lambda_{LM} \sum_{i=1}^{|e|} log \, p_{LM}(e_i|e_1 \ldots e_{i-1})) \end{aligned} \tag{2.8}$$

In order to accommodate many-to-many relationships word alignments (Brown et al., 1993; Dempster et al., 1977) of both directions are taken, i.e. from English to foreign and from foreign to English. The intersection of these alignments give alignment points with a high level of precision, conversely their union gives high recall. To

achieve a good balance between the two the intersection is taken and the missing alignment points are grown heuristically (Och and Ney, 2003). From these word alignments phrase pairs are extracted that are consistent with the alignment. With these phrase pairs the translation probabilities are estimated with maximum likelihood estimations:

$$\phi(\bar{f}|\bar{e}) = \frac{count(\bar{f},\bar{e})}{count(\bar{e})} \tag{2.9}$$

2.4.1 Language Models

The probability of an English sentence $p(e)$ is calculated using a statistical language model. The English sentence e is represented by the sequence of words $e_1, e_2, \ldots, e_{l_e}$ and its probability is decomposed using the chain rule:

$$p(e) = p(e_1)p(e_2|e_1)\ldots p(e_{l_e}|e_1, e_2, \ldots, e_{l_e-1}) \tag{2.10}$$

As the length of the context of a word grows the likelihood of having seen it in the training corpus decreases. In order to accurately estimate the parameters of the model we employ the Markov assumption which says that the probability of a certain sequence can be well estimated from a limited history. Typically the two previous words in a sentence are used forming a trigram language model:

$$p(e_1)p(e_2|e_1)\ldots p(e_{l_e}|e_1, e_2, \ldots, e_{l_e-1}) \approx p(e_1)p(e_2|e_1)\ldots p(e_{l_e}|e_{l_e-2}, e_{l_e-1}) \tag{2.11}$$

$$p(e) \approx \prod_{i=1}^{l_e} p(e_i|e_{i-2}, e_{i-1}) \tag{2.12}$$

The probabilities are estimated via maximum likelihood estimations, these estimates are usually smoothed to ensure all possible sequences have a non-zero probability:

$$p(e_3|e_1, e_2) = \frac{count(e_1, e_2, e_3)}{count(e_1, e_2)} \tag{2.13}$$

As we are transliterating rather than translating our data is likely to be much less sparse allowing for language models of character sequences to be built of higher orders, that accurately estimate longer n-gram sequences, than what is typical in SMT.

2.4.2 Optimisation & Evaluation

To be able to achieve optimal performance the weights of the model ($\lambda_\phi, \lambda_{LM}$) need to be tuned. This is done by setting aside a development corpus, separate to the training and test corpora and altering the weights until the model achieves the optimal performance translating the development corpus.

The major performance metric used in SMT is Bleu (Papineni et al., 2001) an automatic metric that has been seen to correlate well with human judgements of translation quality. This computes the number of common n-grams between a candidate translation and one or more reference translations.

Minimum error rate training (Mert) is used to automatically tune the weights (Och, 2003). The system generates the top n best translations, these are scored with Bleu then possible feature weights are searched for that move good translations higher and bad translations lower. The model is altered accordingly and the process repeats until optimal performance is achieved.

An alternative method of learning phrase translations is the Joint Probability Model (Marcu and Wong, 2002) which instead of extracting phrases with the aid of a heuristic, directly estimates the joint probability of phrase translations from a parallel corpus. Moving from word to phrase alignments greatly increases the time needed to train the model, however this method has been shown to outperform heuristic approaches on reduced data sets.

The next chapter describes the baseline system and our approach to transliteration using Moses.

Chapter 3

Approach

This chapter describes our baseline transliteration system and how transliteration is accomplished with Moses.

3.1 Baseline

The baseline systems' character alignments for English-Chinese and Arabic-English roughly follow the average length ratios (see Sections 4.1.1 and 4.1.2) in the respective corpora of 2 English characters to every 1 Chinese and 1 English character to every 1 Arabic. After aligning the characters the longer of the two lists of characters is truncated to match the length of the shorter.

Characters are transliterated via the most frequent mapping found in each training corpora with the exception of English to Chinese where the English name is split, as in training, into pairs of characters and transliterated along with any trailing single characters. Any unknown character or pair of characters is transliterated as the empty string.

3.2 Transliterating with Moses

Moses offers a more principled method of both learning useful segmentations and combining them in the final transliteration process. Segmentations or phrases are learnt by taking intersection of the bidirectional character alignments and heuristically growing missing alignment points, described in Section 2.4. This allows for phrases that better reflect segmentations made when the name was originally transliterated than the naïve method used in our baseline.

Having learnt useful phrase transliterations and built a language model over the target side characters, these two components are given weights and combined during the decoding of the source name to the target name. Decoding builds up a transliteration from left to right and since we are not allowing for any reordering the foreign characters to be transliterated are selected from left to right as well, computing the probability of the transliteration incrementally.

Decoding proceeds as follows:

- Start with no foreign characters having been transliterated, this is called an empty hypothesis, we then expand this hypothesis, to make other hypotheses covering more characters
- A foreign phrase $\bar{f}_i$ to be transliterated into an English phrase $\bar{e}_i$ is picked, this phrase must start with the left most character of our foreign name that has yet to be covered, potential transliteration phrases are looked up in the translation table
- The evolving probability is computed as a combination of language model, looking at the current character and the previously transliterated $n-1$ characters, depending on n-gram order, and transliteration model probabilities

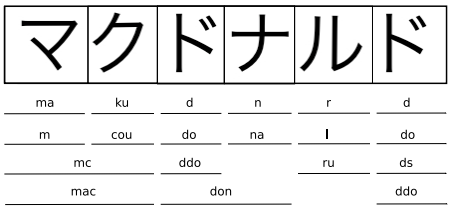

Figure 3.1: Example Transliteration Options

Each hypothesis stores information on what foreign characters have been transliterated so far, the transliteration of the hypothesis' expansion, the probability of the transliteration up to this point and a pointer to its parent hypothesis. The process of hypothesis expansion continues until all hypotheses have covered all foreign characters. The chosen hypothesis is the one which covers all foreign characters with the highest probability. The final transliteration is constructed by backtracking through the parent nodes in the search that lay on the path of the chosen hypothesis (Equation 2.8).

To search the space of possible hypotheses exhaustively is unfeasible and Moses employs a number of techniques to reduce this search space, some of which can lead to search errors.

One advantage of using a Phrase-based SMT approach over previous more linguistically informed approaches (Knight and Graehl, 1997; Stalls and Knight, 1998; Al-Onaizan and Knight, 2002) is that no extra information is needed other than the surface form of the name pairs. This allows us to build transliteration systems in languages that do not have such information readily available and cuts out errors made during intermediate processing of names to say a phonetic or romanized representation. However only relying on surface forms for information on how a name is transliterated misses out on any useful information held at a deeper level.

The next chapter gives the details of the data sets, software and metrics used as well as descriptions of the experiments.

Chapter 4

Experimental Framework

This chapter discusses the data sets used to train and test the English-Chinese and Arabic-English transliteration models, the software used throughout the project, the metrics used to in evaluation and finally gives descriptions of our experiments.

4.1 Data

The following sections describe the format and preprocessing of both the English-Chinese and Arabic-English data. The final data sets used for the Chinese and Arabic experiments were made as similar as possible in order to better compare their performance.

After preprocessing, each corpora was split into training, development and test sets. The development and test sets were formed by randomly selecting 500 name pairs, the training set being made up of the name pairs that were left. There was no attempt to remove unbalanced punctuation, i.e. punctuation that is only present in one of the names in a name pair.

The final English-Chinese corpora was assumed to contain just English names and their Chinese transliterations and the Arabic-English corpora was assumed to contain just Arabic names and their English transliterations.

4.1.1 Chinese

- **Corpus name:** Chinese <–> English Name Entity Lists v 1.0 (LDC2005T34)[1], (`ldc_propernames_people_ec_v1.beta.txt`)

[1] Linguistic Data Consortium catalog number

Knute	/克努特/
Knutel	/克尼特尔/
Knuth	/克努特/
Knuthsen	/克努森/
Knutov	/克努托夫/
Knuts	/克努茨/
Knutsen	/克努森/
Knutson	/克努森/克努特松/

Figure 4.1: Sample English-Chinese Data

k n u t e	克 努 特
k n u t e l	克 尼 特 尔
k n u t h	克 努 特
k n u t h s e n	克 努 森
k n u t o v	克 努 托 夫
k n u t s	克 努 茨
k n u t s e n	克 努 森
k n u t s o n	克 努 森
k n u t s o n	克 努 特 松

Figure 4.2: Sample English-Chinese Preprocessed Data

- **Encoding:** GB-2312
- **Script:** Simplified Chinese
- **Number of English names:** 572213
- **Chinese transliterations:** 673385
- **Average number of English characters per name:** 6.08
- **Average number of Chinese characters per name:** 2.87

Preprocessing was as follows:

- Encoding converted from GB-2312[2] to UTF-8[3] for ease of processing and viewing[4]
- Multiple transliterations were flattened, forward slashes removed
- English lowercased
- Both English names and Chinese transliterations space-delimited
- The intersection of the flattened list and the 1990 US Census name lists[5], comprising of surnames as well as female and male first names was taken, leaving 78251 name pairs. This was done to create a subset of common English names as the original contained names from many different countries.
- Alternative *translations* of Chinese, Japanese and Korean names where left intact.

4.1.2 Arabic

- **Corpus name:** 10001 Arabic Names (LDC2005G02)[6]
- **Encoding:** Standard Arabic Technical Transliteration System (SATTS)
- **Number of English names:** 11367

[2]Standardization Administration of China website: `http://www.sac.gov.cn/english/home.asp`
[3]Unicode Home website: `http://www.unicode.org/`
[4]Conversion tool: `http://www.mandarintools.com/zhcode.html`
[5]US Census name lists website: `http://www.census.gov/genealogy/names/names_files.html`
[6]Linguistic Data Consortium catalog number

DAF"I	Dafi'i (s)
DAFL	Dafil (s)
DAFID	David
DAQW	Daqu (s)
DAQI;	Daqiz (s)
DAK:I	Dakishi (s)

Figure 4.3: Sample Arabic-English Data

d a f i ' i	د ا ف ع ي
d a f i l	د ا ف ل
d a v i d	د ا ف ي د
d a q u	د ا ق و
d a q i z	د ا ق ي ز
d a k i s h i	د ا ك ش ي

Figure 4.4: Sample Arabic-English Preprocessed Data

- **Arabic transliterations:** 11367
- **Average number of English characters per name:** 14.06
- **Average number of Arabic characters per name:** 15.40

Preprocessing was as follows:

- Encoding converted from SATTS to UTF-8 for ease of processing and viewing
- English lowercased and extraneous information removed
- Both Arabic names and English transliterations space-delimited and their order switched

4.2 Software

The following sections describe briefly the software that was used during the project.

4.2.1 Moses

Moses (Koehn et al., 2007) is an SMT system that allows you to automatically train translation models for any language pair. All you need is a collection of translated texts (parallel corpus).

- beam-search: an efficient search algorithm that quickly finds the highest probability translation among the exponential number of choices
- phrase-based: the state-of-the-art in SMT allows the translation of short text chunks
- factored: words may have factored representation (surface forms, lemma, part-of-speech, morphology, word classes...)[7]

Available from: `http://www.statmt.org/moses/`

[7] Taken from website

4.2.2 GIZA++

GIZA++ (Och and Ney, 2003) is an extension of the program GIZA (part of the SMT toolkit EGYPT) which was developed by the Statistical Machine Translation team during the summer workshop in 1999 at the Center for Language and Speech Processing at Johns-Hopkins University (CLSP/JHU).[8] GIZA++ extends GIZA's support to train the IBM Models (Brown et al., 1993) to cover Models 4 and 5. GIZA++ is used by Moses to perform word alignments over parallel corpora.

Available from: `http://www.fjoch.com/GIZA++.html`

4.2.3 SRILM

SRILM (Stolcke, 2002) is a toolkit for building and applying statistical language models (LMs), primarily for use in speech recognition, statistical tagging and segmentation.[8] SRILM is used by Moses to build statistical language models.

Available from: `http://www.speech.sri.com/projects/srilm/`

4.2.4 Mert

Ashish Venugopal's implementation of Mert for SMT as described in Och (2003) and Venugopal and Vogel (2005). It includes several improvements to the basic training method including randomized initial conditions and permuted model order (in order to deal with the greedy nature of the algorithm) and dynamic parameter range expansion or restriction (to increase their potential relative impact, or in order to limit the use of certain models).[8] Mert is used by Moses to optimise performance.

Available from: `http://www.cs.cmu.edu/~ashishv/mer.html`

4.2.5 Bleu

Bleu (Papineni et al., 2001) is an automatic Machine Translation (MT) evaluation metric that has been shown to correlate well with human judgements on both fluency and adequacy. Bleu is used by Moses during Mert weight optimisation.

[8] Taken from website

Available from: http://www.nist.gov/speech/tests/mt/scoring/index.htm

4.2.6 Python

The baseline system, the script to calculate Levenshtein distance (Levenshtein, 1966) between two strings was implemented in Python as were all other pre and post-processing scripts.

Available from: http://www.python.org/download

4.3 Metrics

Resources were not available to include human judgements in our evaluation of performance, instead the following automatic metrics were used:

- **Bleu (4-Gram):** the current standard SMT metric used in optimising performance with Mert and evaluating systems. Calculates the geometric mean of the precision of n-grams of length 1, 2, 3 and 4 between the candidate transliteration and the reference, weighted to penalise shorter transliterations.

- **Top 1, Top 5, Top 20:** the percentage of correct transliterations in the top 1, top 5 and top 20 candidates.

- **Levenshtein Distance 1:** the percentage of the highest ranked transliterations that have a Levenshtein distance of 1 or 0, i.e. the best scoring transliteration is 0 or 1 insertions, deletions or substitutions away from the reference transliteration.

The Top 1 results give a measure of the performance in the context of an SMT system. As discussed earlier only exact transliterations will help SMT performance as measured by automatic metrics that base scores on n-gram matches between a candidate translation and reference translations. The Top 5 and 20 results show the potential to boost Top 1 accuracy with reranking.

As transliterating is inherently ambiguous, Al-Onaizan and Knight (2001) report human accuracy transliterating Arabic at 75.3%, Levenshtein Distance 1 gives a rough indication of performance when allowing some variation in transliteration.

4.4 Experiments

This section describes our transliteration experiments and their motivation.

4.4.1 Baseline

All the baseline experiments were conducted using all of the available training data and evaluated over the test set using Bleu, Top 1 and Levenshtein 1 metrics. The Top 5 and Top 20 metrics are not applicable as the baseline system only produces one answer.

4.4.2 Default Settings

Experiments varying the length of reordering distance and using Moses' different alignment methods: intersection, grow, grow diagonal and union, gave no change in performance. Monotone translation and the grow-diag-final alignment heuristic were used for all further experiments.

These were the default parameters and data used during the training of each experiment unless otherwise stated:

- **Transliteration Model Data:** All
- **Maximum Phrase Length:** 3
- **Language Model Data:** All
- **Language Model N-Gram Order:** 3
- **Language Model Smoothing & Interpolation:**
  Kneser-Ney (Kneser and Ney, 1995), Interpolate (English-Chinese Forward)
  Automatically Disabled, Interpolate (English-Chinese Back)
  Automatically Disabled, Interpolate (Arabic-English Forward)
  Automatically Disabled, Interpolate (Arabic-English Back)
- **Alignment Heuristic:** grow-diag-final
- **Reordering:** Monotone
- **Maximum Distortion Length:** 0

- **Model Weights:**
  - **Translation Model:** 0.2, 0.2, 0.2, 0.2, 0.2
  - **Language Model:** 0.5
  - **Distortion Model:** 0.0
  - **Word Penalty:** -1

An independence assumption was made between the parameters of the transliteration model and their optimal settings were searched for in isolation. The best performing settings over the development corpus were combined in the final evaluation systems.

4.4.3 Transliteration Model

The following two experiments were performed to investigate the effect of the transliteration model on performance:

- **Maximum Phrase Length:** increasing the maximum length of phrases entered into the phrase table. Learning longer phrases allows for potentially more accurate transliteration as large phrases can be reused unchanged and the effects of local context on transliteration can be learnt.
- **Transliteration Model Size:** doubling the number of name pairs used in training the transliteration model. The more data used in training the more accurately the phrase translation probabilities can be estimated. Using more data also allows for longer phrases to be learnt as they will tend to increase in frequency the more data is looked at, surpassing the minimum number of occurrences needed by Moses to be considered for entry into the phrase table.

4.4.4 Language Model

The following two experiments were performed to investigate the effect of the language model on performance:

- **Language Model N-Gram Order:** varying the size of the context used in estimating the language model. This experiment will find the best performing n-gram size with which to estimate the target character language model with a given amount of data.

- **Language Model Size:** doubling the amount of the target language model's training data. Increasing the amount the language model's training data should allow it to make more accurate estimations and help rule out malformed names.

- **Gigaword Proper Noun Language Model:** using a language model built over the characters of the proper nouns (NNP, NNPS) from the parsed New York Times Newswire Service section of the English Gigaword corpus, comprised of some 700M proper nouns. This experiment will show if any improvement to modelling English names can be made using a vastly larger set of names to approximate the English character language model.

6,7,8,9 and 10-gram models (Kneser-Ney smoothing, interpolated) were built for the Gigaword experiments and the model weights were optimised using Mert, using the best performing settings found in the previous experiments. 6-gram models were found to be the best performing language models in the previous experiments varying n-gram order using only the English data from the name pairs.

4.4.5 Optimising with Mert

- **Optimising Weights with Mert:** optimising the model's weights running Mert over the 500 name pairs of in the development corpus. Mert uses Bleu 4 as a scoring metric as such Chinese names less than 4 characters long were scoring zero. A dummy character was added to the end of each name regardless of length to overcome this.

Although we would like to optimise performance over the Top 1 accuracy to maximise the potential of our method to improve SMT performance, using this metric poses a problem when the reference transliteration does not appear in the top n list of candidates during Mert optimisation. If the reference transliteration does not appear in the n best list then all the candidates have a Top 1 score of 0 and Mert will not be able to optimise the weights any further. It is reasonable to assume, given even a poorly designed system, not all of the top n candidates will score zero using Bleu and so not stop Mert from continuing to optimise the model's weights. Also as Bleu is a measure of common n-grams between reference and candidate transliterations maximising performance over Bleu is assumed to increase the performance of the other metrics; Top 1, 5, 20 and Levenshtein 1.

The next chapter gives the results of the transliteration experiments.

Chapter 5

Results

In this chapter we present the results of the experiments described in the previous chapter. Initial experiments to find the best parameters and to investigate the effects of increasing training data were run over the 500 name pairs in the development corpus as was Mert in order to tune the model's weights. The test corpus only being used to report final results.

5.1 Chinese Results

5.1.1 Baseline & Default Settings

Table 5.1 gives the results transliterating forwards and back to and from Chinese using the baseline system as detailed in Section 3.1 and the default settings and weights for Moses as detailed in Section 4.4.2. These results show that even without exploring Moses' parameters or tuning of the weights it is already a more productive way of transliterating when compared to our baseline.

	Bleu	**Top 1**	**Top 5**	**Top 20**	**Levenshtein 1**
Baseline Forward (test)	15.2	10.2	n/a	n/a	33.6
Default Settings Forward (test)	42.3	30.0	48.4	67.2	64.6
Baseline Back (test)	30.9	4.2	n/a	n/a	21.6
Default Settings Back (test)	54.1	17.6	35.6	54.4	48.0

Table 5.1: English-Chinese Baseline & Default Settings

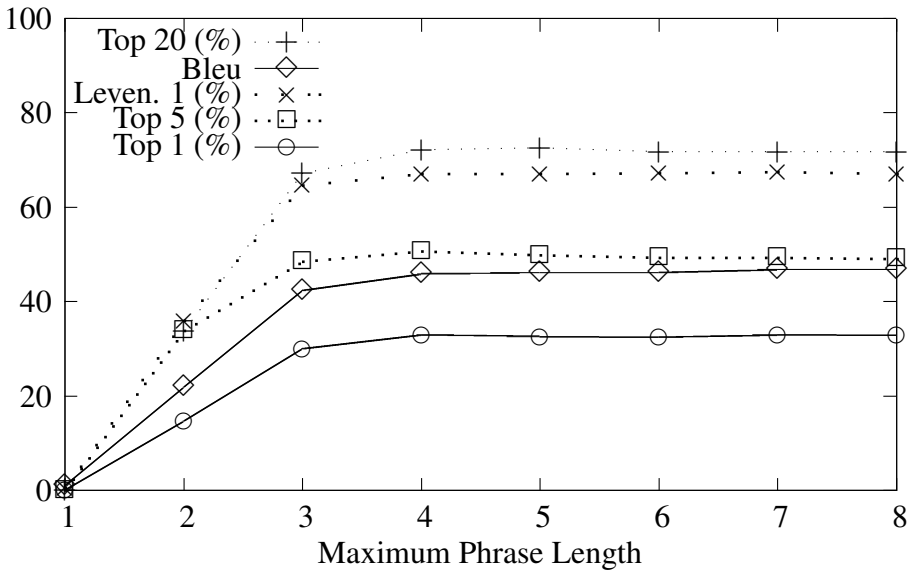

Figure 5.1: English-Chinese Maximum Phrase Length (Forward)

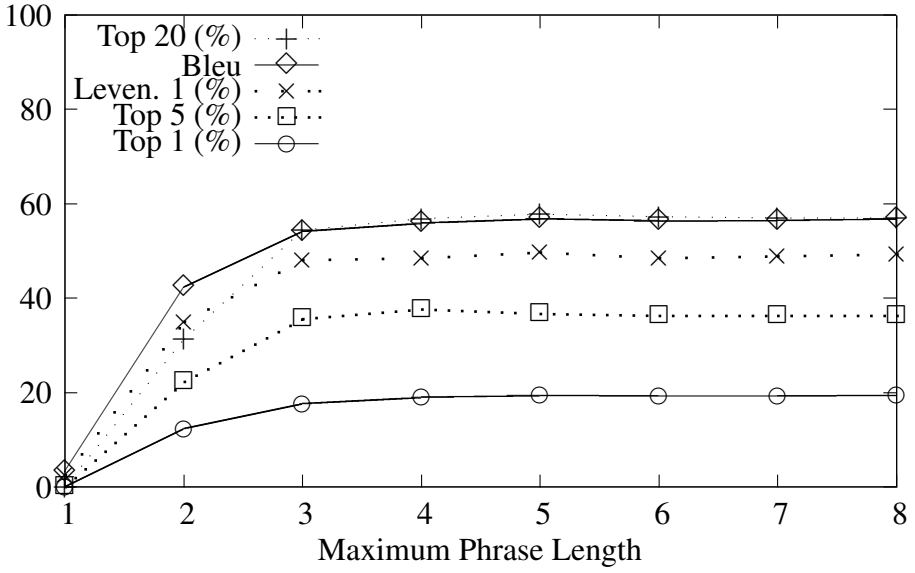

Figure 5.2: English-Chinese Maximum Phrase Length (Back)

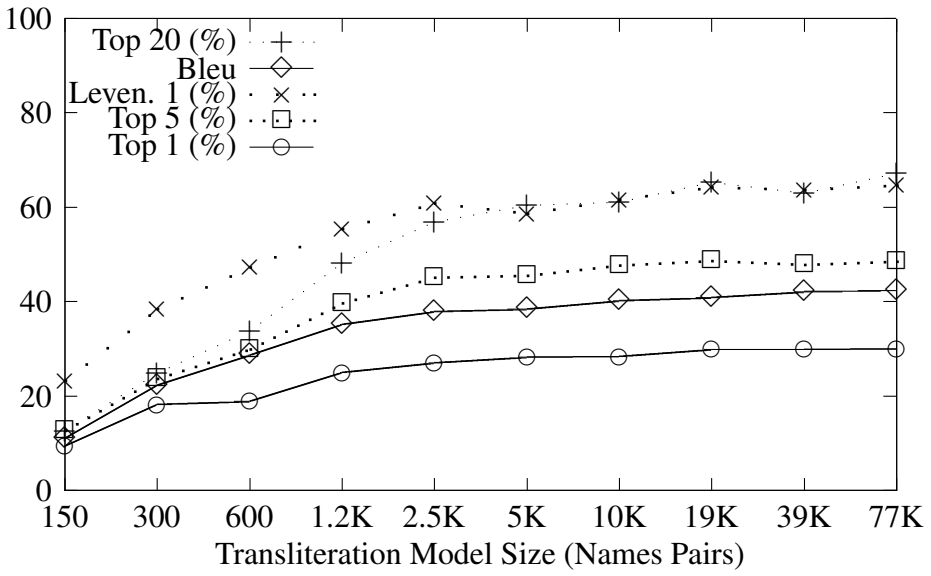

Figure 5.3: English-Chinese Transliteration Model Training Data Size (Forward)

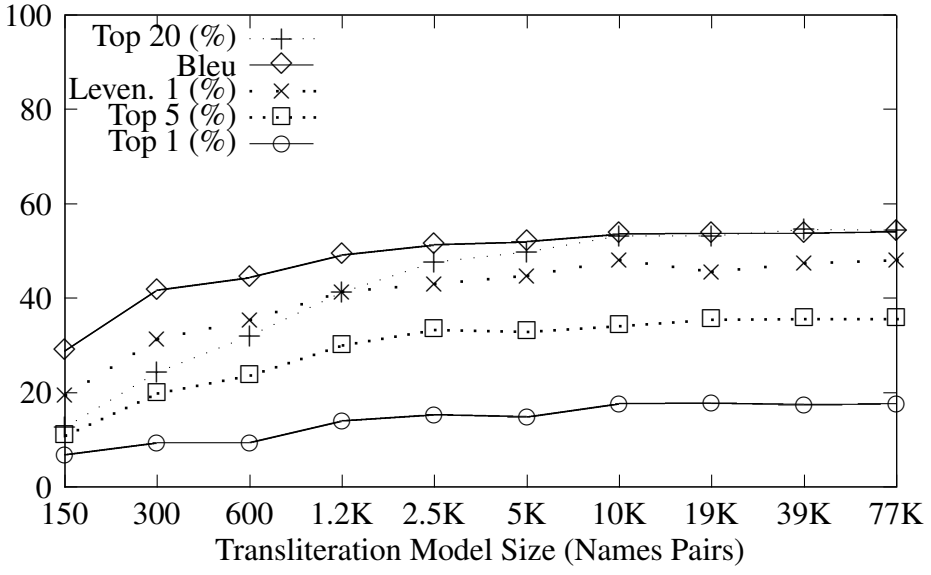

Figure 5.4: English-Chinese Transliteration Model Training Data Size (Back)

5.1.2 Transliteration Model

Figure 5.1 shows the results of transliterating English names into Chinese with increasing maximum phrase length. Top 1 performance peaks at maximum phrase length of 4 reflecting the short length of the Chinese transliterations. Figure 5.2 shows the results of transliterating Chinese transliterations back into the original English with increasing maximum phrase length. Top 1 performance peaks at maximum phrase length of 5.

Figure 5.3 shows the results of transliterating English names into Chinese with an increasing number of name pairs. Figure 5.4 shows the results of transliterating Chinese transliterations back into the original English with an increasing number of name pairs. The best performance in both directions comes from using all available name pairs, compare this with the same experiment with Arabic-English data (5.2.2) where we find that performance suffers using all available name pairs.

With a larger number of characters being used to transliterate, 2492 for English to Chinese compared with 28 for Arabic to English (a-z plus ' and - characters), noise in the data through either mistransliterations, spelling mistakes or unbalanced punctuation finds it harder to skew transliterations through badly aligned phrases because any noise will tend to align itself more evenly over the greater number of characters and phrases.

5.1.3 Language Model

Figure 5.5 shows the results of transliterating English names into Chinese with increasing language model n-gram order. The best Top 1 performance comes with a 2-gram language model reflecting the short average length of the Chinese transliterations. Figure 5.6 shows the results of transliterating Chinese transliterations back into the original English with increasing language model n-gram order. Top 1 performance peaks with a 6-gram model using the set of 77K names, higher orders give a slight decrease in performance.

Figure 5.7 shows the results of transliterating English names into Chinese with an increasing amount of data used to estimate the Chinese character language model. Figure 5.8 shows the results of transliterating Chinese transliterations back into the original English with an increasing amount of data used to estimate the English character language model. Both directions give the best performance using the maximum number of names.

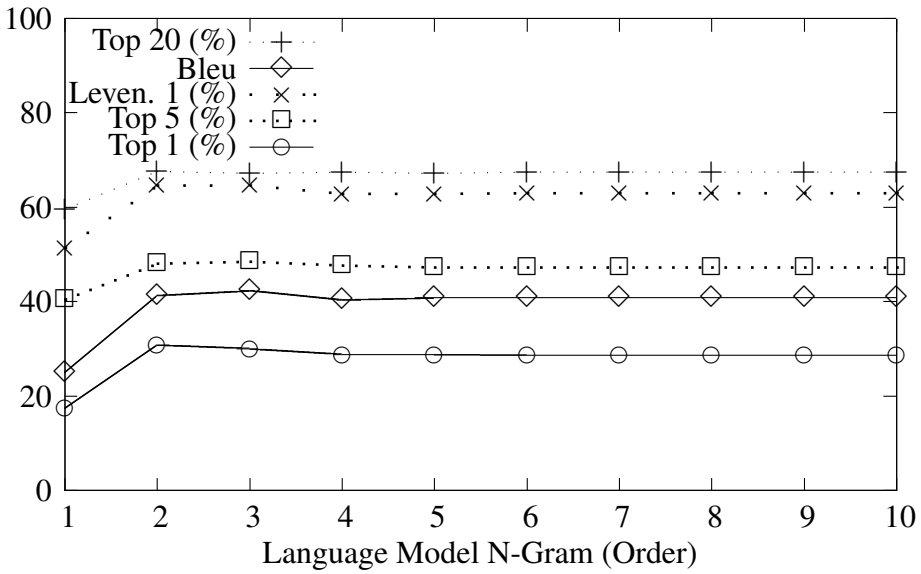

Figure 5.5: English-Chinese Language Model N-Gram Order (Forward)

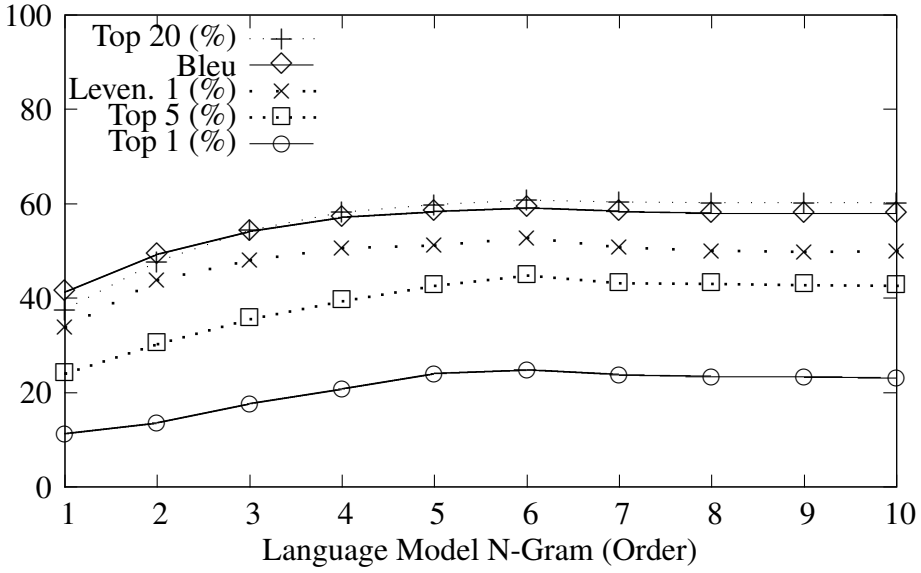

Figure 5.6: English-Chinese Language Model N-Gram Order (Back)

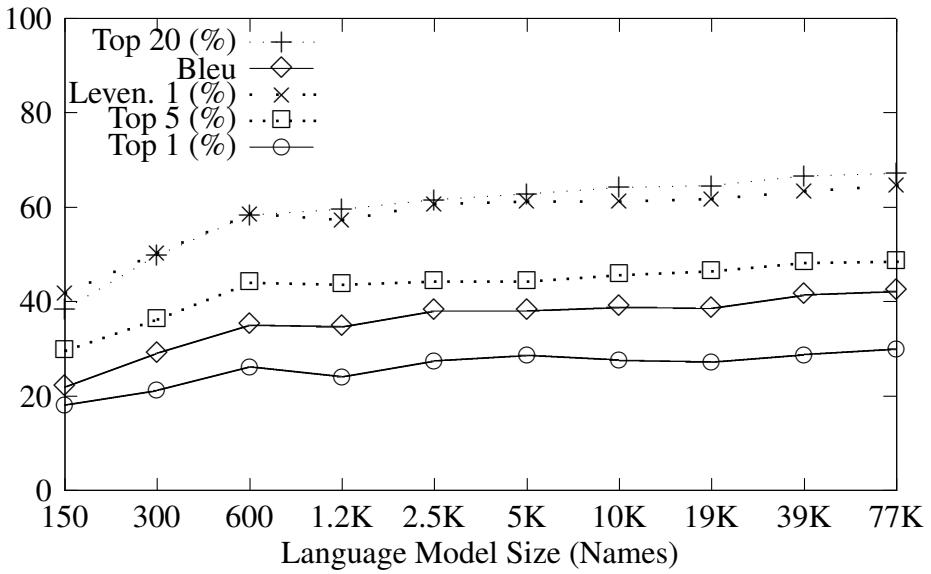

Figure 5.7: English-Chinese Language Model Training Data Size (Forward)

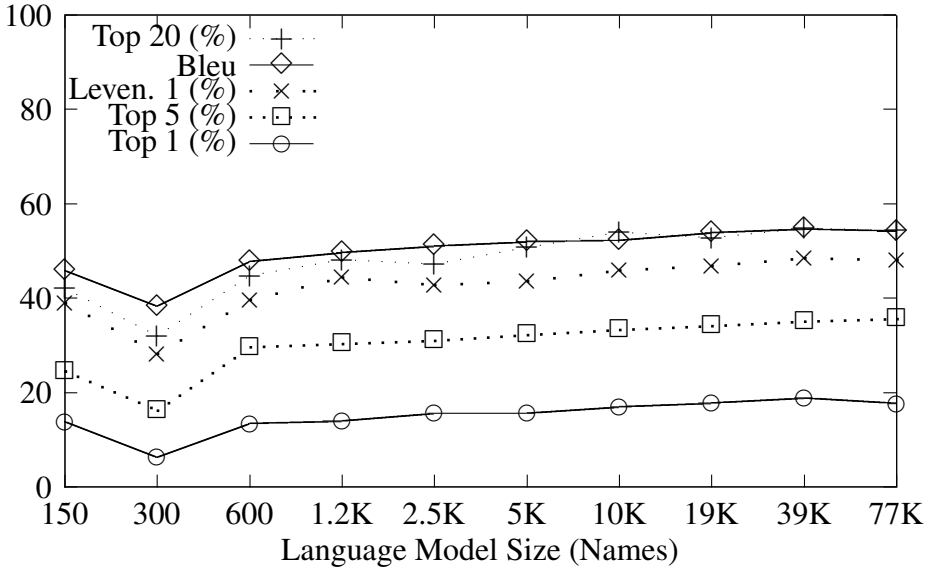

Figure 5.8: English-Chinese Language Model Training Data Size (Back)

Table 5.4 gives the results with using language models built over 700M proper names from the New York Times section of the Gigaword Corpus. Using a 6-gram model built with the Gigaword proper nouns we see a big jump in performance over the language model built from the original set of 77K names. Whereas performance plateaued pushing the n-gram order higher than 6 with the original 77K names, building higher order models over 700M names gives Top 1 performance increases up to 9-gram models.

5.1.4 Optimising with Mert

Tables 5.2 and 5.3 gives the results using the best performing parameter settings found in the previous experiments using the defaults weights given in Section 4.4.2 as well as the results using the weights obtained using Mert. The actual parameters and weights used are given below.

Best performing *forward* transliteration settings and Mert weights:

- **Maximum Phrase Length:** 4
- **Transliteration Model Data:** 77K names
- **N-Gram Order:** 2
- **Language Model Data:** 77K names
- **Mert Model Weights:**
  - **Translation Model:** 0.112487, 0.046433, 0.165974, 0.018357, 0.064108
  - **Language Model:** 0.175834
  - **Word Penalty:** 0.200436

Best performing *back* transliteration settings and Mert weights:

- **Maximum Phrase Length:** 5
- **Transliteration Model Data:** 77K names
- **N-Gram Order:** 10

	Bleu	Top 1	Top 5	Top 20	Levenshtein 1
Default Settings & Weights (test)	42.3	30.0	48.4	67.2	64.6
Best Settings, Default Weights (dev)	43.3	31.8	49.6	72.0	67.2
Best Settings, Default Weights (test)	46.7	31.4	53.6	69.8	67.0
Best Settings, Mert Weights (test)	58.2	37.8	59.8	77.2	72.2

Table 5.2: English-Chinese Forward Transliteration Final System

	Bleu	Top 1	Top 5	Top 20	Levenshtein 1
Default Settings & Weights (test)	54.1	17.6	35.6	54.4	48.0
Best Settings, Default Weights (dev)	61.2	26.4	44.8	64.0	54.6
Best Settings, Default Weights (test)	58.3	23.6	36.4	57.8	49.4
Best Settings, Mert Weights (test)	60.5	22.4	44.2	64.2	51.6

Table 5.3: English-Chinese Back Transliteration Final System

	Bleu	Top 1	Top 5	Top 20	Levenshtein 1
77K Names 6-Gram LM (test)	60.5	22.4	44.2	64.2	51.6
Gigaword NNP 6-Gram LM (test)	66.0	29.6	54.6	70.6	59.2
Gigaword NNP 7-Gram LM (test)	67.6	32.0	56.2	72.0	60.4
Gigaword NNP 8-Gram LM (test)	68.2	34.6	55.6	73.8	60.8
Gigaword NNP 9-Gram LM (test)	68.2	34.8	53.4	73.6	61.4
Gigaword NNP 10-Gram LM (test)	69.0	34.8	53.8	73.0	61.8

Table 5.4: English-Chinese Back Transliteration Gigaword Language Model

- **Language Model Data:** 700M Gigaword NYT Proper Nouns

- **Mert Model Weights:**

  - **Translation Model:** 0.097296, 0.040128, 0.101328, 0.042656, 0.096012
  - **Language Model:** 0.056984
  - **Word Penalty:** -0.215401

5.1.5 Example Transliterations

Table 5.5 gives examples of forward transliterations of English names into Chinese. The baseline system gives the correct answer for both of the shorter examples, Julian and Sayer, where the transliterations happen to follow our baseline's approach to segmenting and transliteration. Moses correctly transliterates the longer name Corrington. Longer names are where we'd expect Moses to outperform our baseline with its more intelligent method of learning variable length phrases and using a character language model to help find the correct transliteration in context.

Original	Reference	Baseline	Moses Best System
Julian	尤利安	尤利安	尤利安
Corrington	科林顿	科里因格顿	科林顿
Sayer	萨耶尔	萨耶尔	塞尔

Table 5.5: Example English-Chinese Forward Transliterations

Table 5.6 gives examples of back transliterations of Chinese transliterations of English names into English. Similar to the example forward transliterations given above the baseline back transliterates the shorter name 雷纳托 (Renato) correctly and Mitch to within a Levenshtein distance of 1 but struggles with the longer example of 贝尔格雷夫 (Belgrave) giving the most frequent transliterations of each Chinese character in isolation.

Original	Reference	Baseline	Moses Best System
米奇	Mitch	Mich	Mitch
雷纳托	Renato	Renato	Renato
贝尔格雷夫	Belgrave	Beelgrreff	Belgrave

Table 5.6: Example English-Chinese Back Transliterations

5.2 Arabic Results

5.2.1 Baseline & Default Settings

Table 5.7 gives the results transliterating forwards and back to and from English using the baseline system and the default settings and weights for Moses. These results show similar improvements with using Moses over the baseline to the English-Chinese experiments.

	Bleu	Top 1	Top 5	Top 20	Levenshtein 1
Baseline Forward (test)	15.5	1.8	n/a	n/a	10.8
Default Weights Forward (test)	51.6	21.2	35.0	51.2	43.8
Baseline Back (test)	27.1	5.6	n/a	n/a	23.4
Default Weights Back (test)	52.1	23.2	37.6	53.2	58.4

Table 5.7: Arabic-English Baseline & Default Settings

5.2.2 Transliteration Model

Figure 5.9 shows the results of transliterating Arabic names into English with increasing maximum phrase length. Top 1 performance is best when using maximum phrase length 4. Figure 5.10 shows the results of transliterating English transliterations back into the original Arabic with increasing maximum phrase length. Top 1 performance peaks at maximum phrase 3.

Figure 5.11 shows the results of transliterating Arabic names into English with an increasing number of name pairs. Figure 5.12 shows the results of transliterating English transliterations back into the original Arabic with an increasing number of

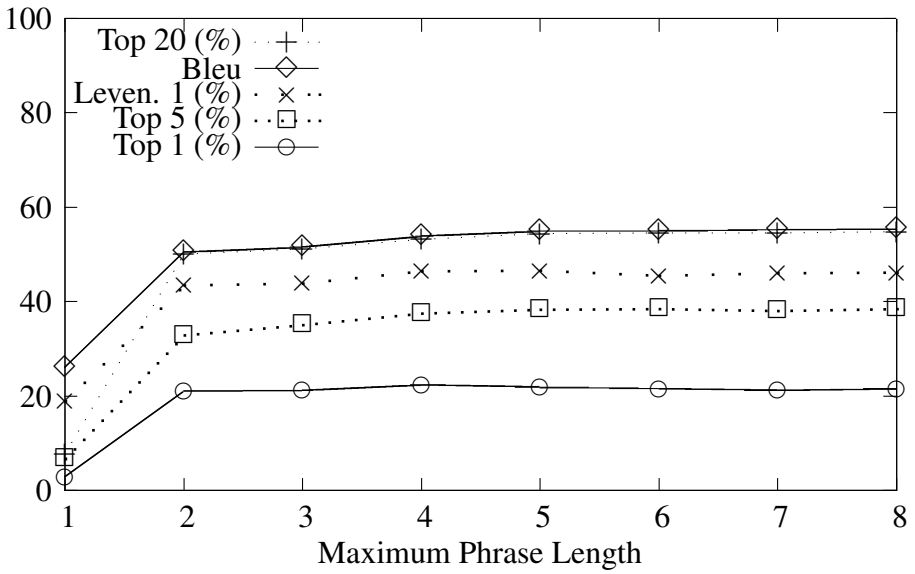

Figure 5.9: Arabic-English Maximum Phrase Length (Forward)

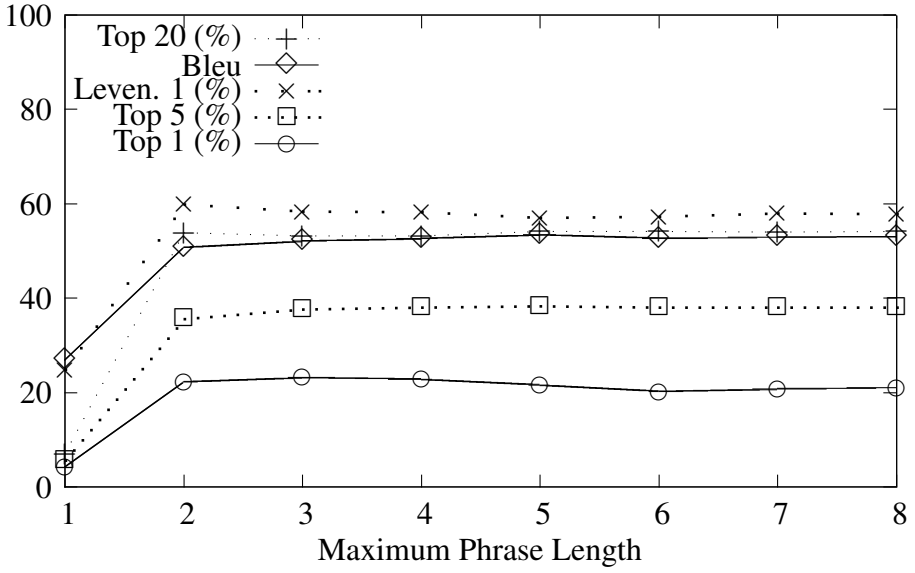

Figure 5.10: Arabic-English Maximum Phrase Length (Back)

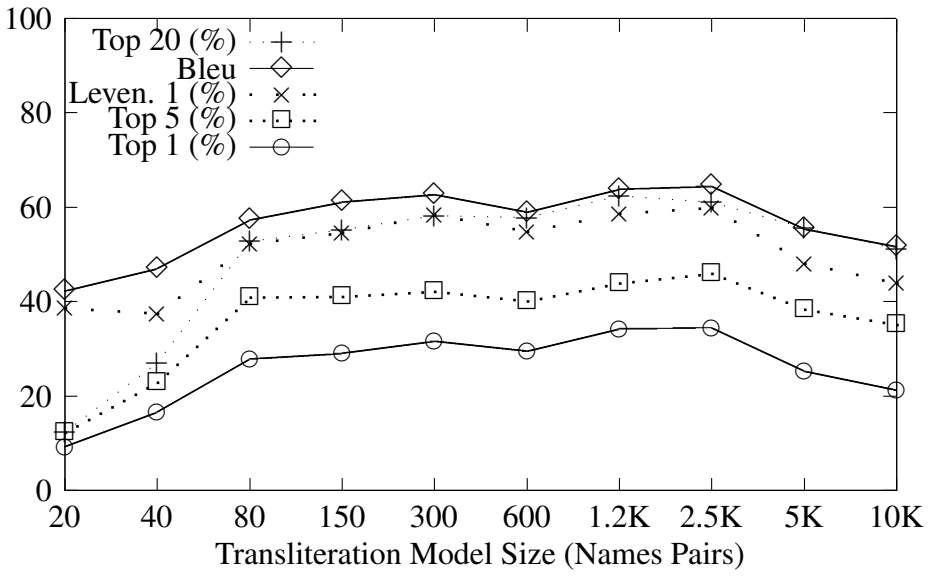

Figure 5.11: Arabic-English Transliteration Model Training Data Size (Forward)

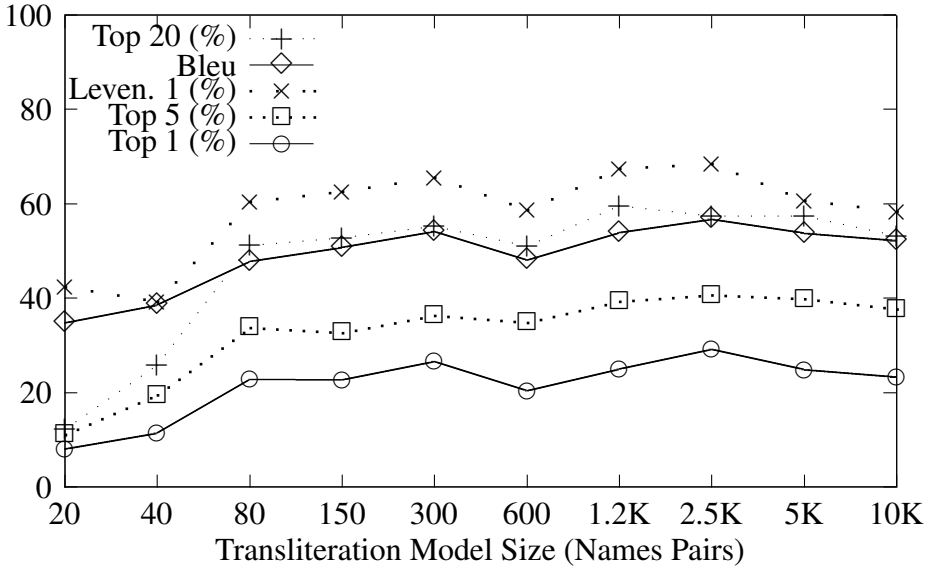

Figure 5.12: Arabic-English Transliteration Model Training Data Size (Back)

name pairs. Performance in both directions suffers using the set of 5K and all 10K names.

With a comparable number of characters used for Arabic names (29) and the English transliterations (28) noise in the data has a greater chance to produce bad phrase pairs and lead to mistransliterations with the increase in training data used for the transliteration model.

5.2.3 Language Model

Figure 5.13 shows the results of transliterating Arabic names into English with increasing language model n-gram order. Best Top 1 performance comes with using a 6-gram English language model. Figure 5.14 shows the results of transliterating English transliterations back into the original Arabic with increasing language model n-gram order. Best Top 1 performance comes with using a 4-gram Arabic language model.

Figure 5.15 shows the results of transliterating Arabic names into English with an increasing amount of data used to estimate the Arabic language model. Figure 5.16 shows the results of transliterating English transliterations back into the original Arabic with an increasing amount of data used to estimate the target side language model. As with the equivalent Chinese experiments (Section 5.1.3) the best performance, in both directions, comes from using all of the language model data.

Table 5.10 gives the results with using a 6-gram language model built over 700M proper names from the New York Times section of the Gigaword Corpus. Using this language model built from English names to model Arabic transliterations hurts performance and no higher n-gram language models were tested.

5.2.4 Optimising with Mert

Tables 5.8 and 5.9 gives the results using the best performing parameter settings found in the previous experiments using the defaults weights given in Section 4.4.2 as well as the results using the weights obtained using Mert. The actual parameters and weights used are given below.

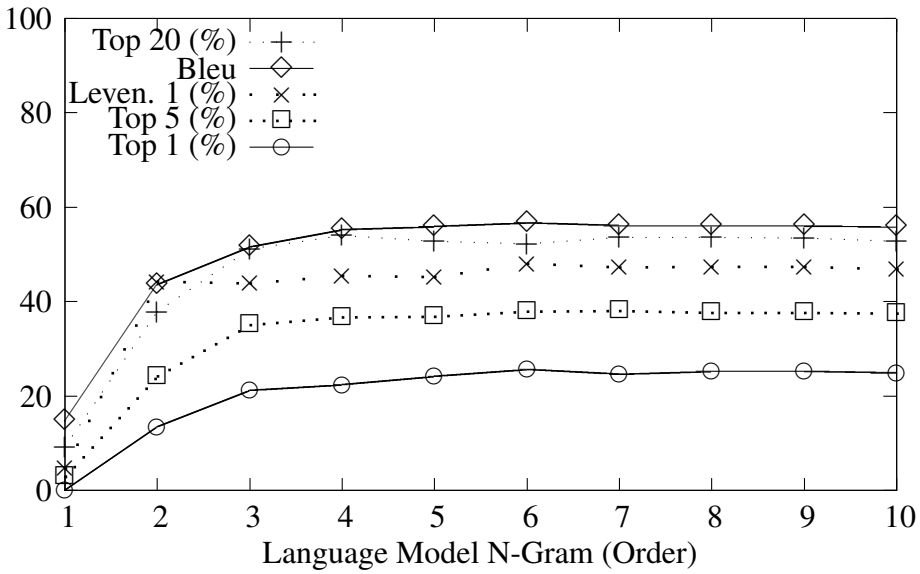

Figure 5.13: Arabic-English Language Model N-Gram Order (Forward)

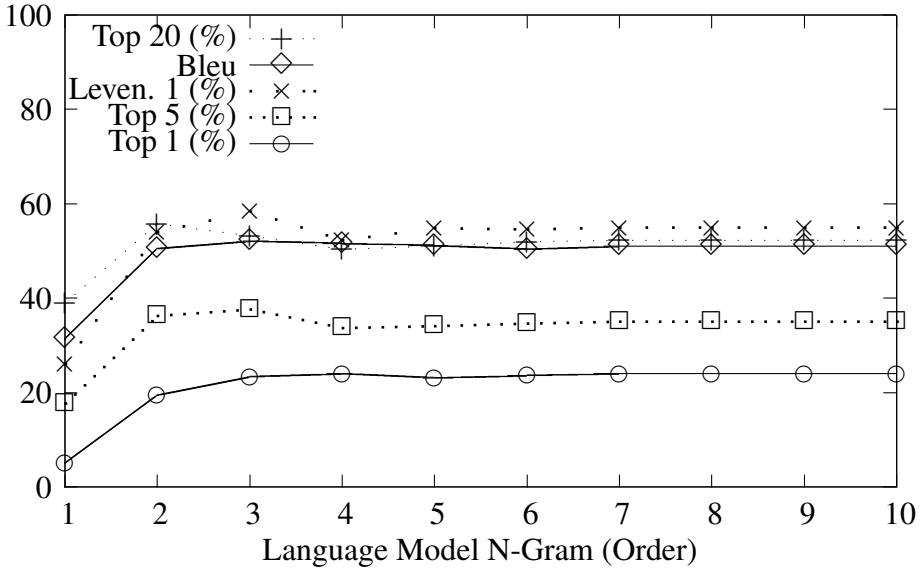

Figure 5.14: Arabic-English Language Model N-Gram Order (Back)

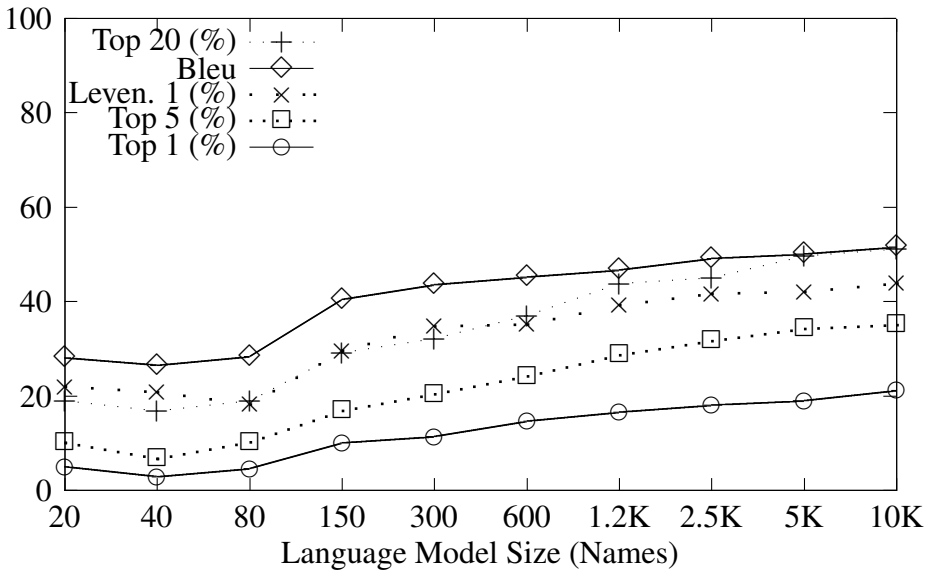

Figure 5.15: Arabic-English Language Model Training Data Size (Forward)

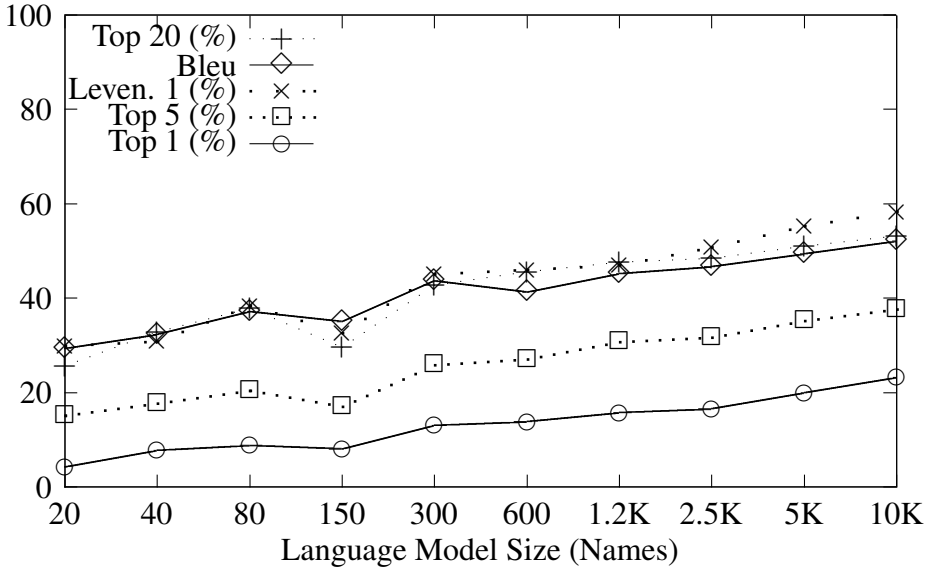

Figure 5.16: Arabic-English Language Model Training Data Size (Back)

Best performing *forward* transliteration settings and Mert weights:

- **Maximum Phrase Length:** 4
- **Transliteration Model Data:** 2.5K name pairs
- **N-Gram Order:** 6
- **Language Model Data:** 10K name pairs
- **Mert Model Weights:**
  - **Translation Model:** 0.081249, 0.081694, 0.116331, 0.006938, 0.216363
  - **Language Model:** 0.107042
  - **Word Penalty:** -0.139256

Best performing *back* transliteration settings and Mert weights:

- **Maximum Phrase Length:** 3
- **Transliteration Model Data:** 2.5K name pairs
- **N-Gram Order:** 4
- **Language Model Data:** 10K name pairs
- **Mert Model Weights:**
  - **Translation Model:** 0.080952, 0.105673, 0.130300, -0.027239, 0.008060
  - **Language Model:** 0.086204
  - **Word Penalty:** 0.204310

5.2.5 Example Transliterations

Table 5.11 gives examples of forward transliterations of Arabic names back into English. The forward transliterations given by the baseline shows similar patterns as the those obtained in the Chinese experiments, the number of shorter names that fit its simpler approach to transliteration being handled correctly but struggling to come up with anything that even closely resembles longer transliterations as can be seen with its transliterations of عبدالرحمني ('Abd-al-Rahmani) and فتحالرحمن (Fath-al-Rahman), which Moses gives correct answers for.

	Bleu	Top 1	Top 5	Top 20	Levenshtein 1
Default Settings & Weights (test)	51.6	21.2	35.0	51.2	43.8
Best Settings, Default Weights (dev)	63.2	35.0	46.0	61.0	53.8
Best Settings, Default Weights (test)	64.6	36.2	47.8	62.2	58.2
Best Settings, Mert Weights (test)	70.3	43.0	58.4	74.6	66.8

Table 5.8: Arabic-English Forward Transliteration Final System

	Bleu	Top 1	Top 5	Top 20	Levenshtein 1
Default Settings & Weights (test)	52.1	23.2	37.6	53.2	58.4
Best Settings, Default Weights (dev)	57.9	31.8	41.0	57.2	64.6
Best Settings, Default Weights (test)	58.4	31.2	40.6	58.2	68.8
Best Settings, Mert Weights (test)	67.1	39.2	60.8	78.4	75.2

Table 5.9: Arabic-English Back Transliteration Final System

	Bleu	Top 1	Top 5	Top 20	Levenshtein 1
Best System, 10K Names LM (test)	70.3	43.0	58.4	74.6	66.8
Best System, Gigaword NNP LM (test)	55.5	26.2	43.4	60.6	53.0

Table 5.10: Arabic-English Forward Transliteration Gigaword Language Model

Original	Reference	Baseline	Moses Best System
عبدالرحمني	’Abd-al-Rahmani	’Bdaaahmna	’Abd-al-Rahmani
مانوك	Manuk	Manuk	Manuk
فتحالرحمن	Fath-al-Rahman	Fthaaahmn	Fath-al-Rahman

Table 5.11: Example Arabic-English Forward Transliterations

Table 5.12 gives examples of back transliterations from English transliterations of Arabic names back into Arabic. Moses gives the correct transliteration of the longer example name Ibn-’Awdiyah, where the baseline gets it wrong but the shorter example Kidudian is correctly transliterated by both systems and the shortest example Burji is only correctly transliterated by the baseline system.

Original	Reference	Baseline	Moses Best System
Kidudian	كيدوديان	كيدوديان	كيدوديان
Ibn-’Awdiyah	ابنعودية	يبنلعايدييا ح	ابنععودية
Burji	بورجي	بورجي	برجي

Table 5.12: Example Arabic-English Back Transliterations

The next chapter concludes and discusses areas of possible future work.

Chapter 6

Conclusion

6.1 Summary

This project has shown that an automatic transliteration system can be built from Phrase-Based SMT system whose performance is comparable to state-of-the-art systems designed specifically to transliterate (Jiang et al., 2007; Zhao et al., 2007). As the construction of a system requires only the surface form of transliterated name pairs, i.e. no additional linguistic information like pronunciations or linguistic constraints, systems for new language pairs can be developed quickly and cheaply with sufficient amounts of data.

Moses outperforms our baseline model in both directions in both English-Chinese and Arabic-English. The best forward transliteration accuracy achieved from English to Chinese was 37.8%, the best back transliteration accuracy from Chinese to English was 34.8%. The best forward transliteration accuracy achieved from Arabic to English was 43.0%, the back transliteration accuracy from English to Arabic was 39.2%.

A significant improvement in transliteration accuracy was made using a higher order language model of English names built over data several orders of magnitude larger in size (34.8% using 700M names) than that used in the basic experiments (23.6% using 77K names).

Moses' performance in the forward transliteration of English to Chinese is comparable to Jiang et al. (2007) who achieved transliteration accuracy of 47.5% for exact matches of the top transliteration candidate and 66% for matches in the top 5 candidates.

Moses' performance in the forward transliteration of Arabic to English is comparable to Zhao et al. (2007) who achieved transliteration accuracy, over a similar 500

name pairs test set, of 29% for exact matches of transliteration candidates and 80% for top candidates with Levenshtein distance of 1 or less.

6.2 Future Work

The degradation in Arabic-English transliteration accuracy with an increasing number of training instances (see Figures 5.11 and 5.12) raises the question of alignment quality and dealing with noise in the data. Possible improvements include handcrafting a character-aligned corpus to either augment or replace a word-aligned corpus whose character alignments are induced automatically. Callison-Burch et al. (2004) found improvements in alignment error rates when using small amounts of word-aligned data in a translation system, it is reasonable to expect similar improvements in a transliteration system would be possible using character-aligned data especially if the amount of word-aligned data available for a given language pair is limited.

Another method for potentially improving alignment quality is using the Joint Probability Model (Marcu and Wong, 2002) to estimate phrase transliteration probabilities. This model has been shown to give improved performance, over the heuristic approach outlined in Section 2.4, with reduced translation data sets and transliteration corpora are small enough for this computationally slow method to be viable.

The big increases in correct transliterations in the top 5 and top 20 lists of candidates show that there is a lot of potential to improve performance by either reranking or a post-processing filter. External features have been shown to be useful to help disambiguate between candidates (Jiang et al., 2007) and these could easily be incorporated in a post-processing step reranking candidates using a Maximum Entropy model (Berger et al., 1996) based on features such as web counts and context of the source name, if used in a translation system.

Using a vastly larger data set to build the English character language model gave great improvements to transliteration accuracy, it would be interesting to see if the improvements could be matched using similar sized data sets in Chinese and Arabic.

Finally Moses allows for preprocessed words to be included in the decoding process or simply copied into a translation directly. It would be interesting to see how much, if at all, Bleu scores for a translation system could be improved using precomputed transliterations of proper names.

Bibliography

AbdulJaleel, N. and Larkey, L. (2003). Statistical transliteration for english-arabic cross language information retrieval. *Proceedings of CIKM*, pages 139–146.

Al-Onaizan, Y. and Knight, K. (2001). Translating named entities using monolingual and bilingual resources. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 400–408.

Al-Onaizan, Y. and Knight, K. (2002). Machine transliteration of names in Arabic text. *Proceedings of the ACL-02 workshop on Computational approaches to semitic languages*, pages 1–13.

Berger, A., Della Pietra, V., and Della Pietra, S. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Brown, P., Della Pietra, V., Della Pietra, S., and Mercer, R. (1993). The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311.

Callison-Burch, C., Talbot, D., and Osborne, M. (2004). Statistical machine translation with word- and sentence-aligned parallel corpora. *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*.

Chen, H., Yang, C., and Lin, Y. (2003). Learning Formulation and Transformation Rules for Multilingual Named Entities. *Proceedings of ACL 2003 Workshop on Multilingual and Mixed-language Named Entity Recognition: Combining Statistical and Symbolic Models*, pages 1–8.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271.

Jiang, L., Zhou, M., Chien, L., and Niu, C. (2007). Named Entity Translation with Web Mining and Transliteration. *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1629–1634.

Kang, B. and Choi, K. (2000). Automatic transliteration and back-transliteration by decision tree learning. *Proceedings of the 2nd International Conference on Language Resources and Evaluation, Athens, Greece*.

Kneser, R. and Ney, H. (1995). Improved backing-off for M-gram language modeling. *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, 1.

Knight, K. and Graehl, J. (1997). Machine Transliteration. *Computational Linguistics*, 24(4):599–612.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al. (2007). Moses: Open source toolkit for statistical machine translation. *Proceedings of the Annual Meeting of the Association for Computational Linguistics, demonstation session*.

Koehn, P., Och, F., and Marcu, D. (2003). Statistical phrase-based translation. *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54.

Levenshtein, V. (1966). Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707.

Marcu, D. and Wong, W. (2002). A phrase-based, joint probability model for statistical machine translation. *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 133–139.

Och, F. (2003). Minimum error rate training in statistical machine translation. *Proceedings of the 41th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 160–167.

Och, F. and Ney, H. (2001). Discriminative training and maximum entropy models for statistical machine translation. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 295–302.

Och, F. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Oh, J., Choi, K., and Isahara, H. (2006). A Comparison of Different Machine Transliteration Models. *Journal of Artificial Intelligence Research*, 27:119–151.

Papineni, K., Roukos, S., Ward, T., and Zhu, W. (2001). BLEU: a method for automatic evaluation of machine translation. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318.

Stalls, B. and Knight, K. (1998). Translating Names and Technical Terms in Arabic Text. *Proceedings of the COLING/ACL Workshop on Computational Approaches to Semitic Languages*, pages 34–41.

Stolcke, A. (2002). SRILM-an extensible language modeling toolkit. *Proceedings of the ICSLP*, 2:901–904.

The Economist Technology Quarterly (2007). What's in a name? *The Economist (The Economist Technology Quarterly)*, 382(8519):27–29.

Venugopal, A. and Vogel, S. (2005). Considerations in Maximum Mutual Information and Minimum Classification Error training for Statistical Machine Translation. *Proceedings of the Tenth Conference of the European Association for Machine Translation (EAMT-05)*.

Virga, P. and Khudanpur, S. (2003). Transliteration of proper names in cross-language applications. *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 365–366.

Zhao, B., Bach, N., Lane, I., and Vogel, S. (2007). A Log-linear Block Transliteration Model based on Bi-Stream HMMs. *Proceedings of NAACL HLT*, pages 364–371.