A Generic Framework for Machine Transliteration

A Kumaran Tobias Kellner
Multilingual Systems Research
Microsoft Research India
Bangalore, INDIA
a.kumaran@microsoft.com

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: CLIR, Machine Transliteration

General Terms

Algorithms

Keywords

Machine Transliteration, Cross-language Information Retrieval

1. INTRODUCTION

Machine Transliteration deals with the conversion of text strings from one orthography to another, while preserving the phonetics of the strings in the two languages. Transliteration is an important problem in machine translation or crosslingual information retrieval, as most proper names and generic iconic terms are out-of-vocabulary words, and therefore need to be transliterated. There are numerous methods explored in the literature for machine transliteration, ranging from rule-based techniques to statistical learning techniques. Here we focus our attention on language-independent techniques that potentially can scale well with a large number of languages. In this paper, we present a modular, statistical learning framework that lends itself for easy experimentation with transliteration tasks between a variety of different languages, in a language-independent manner. The workbench includes a variety of components – algorithms, data-sets and transliterations scripts – for a quick assembly of an effective transliteration system across langauges. We believe that such workbenches would be important in an increasingly multilingual world, for building systems that span a number of languages, quickly and effectively.

2. TRANSLITERATION FRAMEWORK

We present our machine transliteration framework based on a core algorithm modeled as a *noisy channel* [3]; the model poses the transliteration problem as a noisy-channel, where the source string gets *garbled* into target string. The transliteration is learned by estimating the parameters of the distribution that maximizes the likelihood of observing the garbling seen in the training data. Subsequently, given a target language string t, we decode a posteriori the most probable source language string s that gave raise to t, as:

$$s \;=\; arg\ max_s P(s|t) \;=\; arg\ max_s P(t|s)P(s)$$

The transliteration model $P(t|s)$ is learnt from the training corpus and the $P(s)$ is the language model for the source language strings. We segment the source and target strings as $\mathrm{S} = \{s_1 s_2 \ldots s_n\}$ and $\mathrm{T} = \{t_1 t_2 \ldots t_m\}$, where s_i and t_i are source and target language segments respectively, and estimate the $P(t|s)$ approximately as:

$$P(t|s) \;\approx\; \prod_i P(t_i|s_i)$$

Clearly, choosing automatically the right granularity for s_i and t_i without language-specific tools or information, is a major challenge. Hence, we explore an expectation maximization approach, which exploits the only information we know about the alignment, that some prefix (or suffix) of the source string must map to *some* prefix (or suffix, respectively) of the target string, in each of the paired strings in the training set. We extract substrings of up to a specified length from both ends of each of the paired strings, as the initial hypothesis for our segment alignment; the frequencies of such alignments are used for estimating their alignment probabilities. Subsequently, we use viterbi algorithm to find the optimal alignment for each of the paired strings using the estimated alignment probabilities from the first step; the segment alignment probabilities are recalculated in each re-training step, based on the counts of segment pairs in the optimal alignments obtained in that step. At each re-training step, a test set is used to compute the transliteration accuracy, and the training is continued till the point when transliteration accuracy starts decreasing, due to over-fitting. The resulting transliteration model is used subsequently for that specific language pair.

Related Work Our work parallels the language-independent approaches discussed in [5, 1, 4]. These works are primarily based on source-channel models, and their results shown on specific language pairs; in some cases the transliteration was modeled through the phonemic domain. Our framework is closest to that presented in [4], where the transliteration task was modeled as a source-channel [3] based on the orthographic n-grams of the source and target languages. However, we take an additional step making the framework modular and scalable, together with a scripting language for modeling training procedures.

SIGIR'07, July 23–27, 2007, Amsterdam, The Netherlands.
ACM 978-1-59593-597-7/07/0007.

	Exact matches			Fuzzy Matches		
Train Size	Top 1	Top 5	Top 10	Top 1	Top 5	Top 10
500	0.271	0.446	0.503	0.479	0.762	0.729
1,000	0.298	0.493	0.555	0.514	0.814	0.814
2,000	0.326	0.513	0.570	0.535	0.822	0.855
5,000	0.341	0.541	0.601	0.556	0.853	0.878
10,000	0.352	0.565	0.613	0.570	0.882	0.890
20,000	0.353	0.568	0.632	0.573	0.889	0.898

Figure 1: Transliteration on a Language-pair

3. IMPLEMENTATION AND RESULTS

We implemented a generic transliteration framework, in which different algorithmic modules or any pre- or post-processing routines may be integrated easily. In addition, we defined a scripting language that can be used for defining a transliteration task as a script. The languages to be handled and the relevant resource and data files may be read in dynamically. A script engine executes a user-defined script, and as directed, record the statistics during the run.

The quality of a transliteration measured as the fraction of correctly transliterated strings is recorded, parameterized by training size, exact or fuzzy matching (the reference and result strings match *fuzzily* when the edit-distance between them is $\leq$ 20% of the reference string size), recall in the top-n most-probable transliterated strings, etc.

Single Language Pairs Figure 1 provides a sample output of a transliteration task from English to Arabic. The script is written to record the results for various parameters, such as, training data size, exact vs fuzzy, top-n results, etc. From a result as above, several trends may be observed easily: first, that the quality of transliteration ($\approx$ 80% of the English names matched fuzzily with one of the top-10 results, when a training set of about 20,000 name pairs); second, that the transliteration quality improves with the size of the training data, but only asymptotically above 5,000; third, fuzzy matches improve quality significantly, but introduces ambiguity in the results. However, introduction of a post-processing module that can resolve the ambiguity may be employed to boost the accuracy of the results (for example, we tried accessing the index of a search engine with $fuzzy$ strings, which resulted in the correct identification of the transliterated string in vast majority of the cases.

Diverse set of languages Figure 2 provides the results of transliteration experiments on a set of languages, namely, English, Arabic, Japanese and 2 Indic (Hindi and Tamil) languages. In this experiment, we experimented with a specific transliteration algorithm (as presented in Section 2), on transliteration tasks from English to one of the 4 target languages, and in the reverse direction. A template script is used for each task and the results are generated for the exact and fuzzy matches in the top-10 results. The objective of this experiment is to profile the transliteration performance of a given algorithm on a diverse set of language pairs. The results indicate that the algorithm exhibited a varying degree of quality in each language pair: For example, transliteration accuracy from English to Arabic was $\approx$ 90%, to Indic languages were $\approx 70 - 80\%$ and to Japanesewas $\approx$ 42%.

Manual analysis indicate that the low quality of transliterating Indic languages were due to improper segmentation, and that in Japanese was due to the weakly learned model

from English to other languages								
Train Size	Hindi		Tamil		Japanese		Arabic	
	Exact	Fuzzy	Exact	Fuzzy	Exact	Fuzzy	Exact	Fuzzy
500	.244	.571	.202	.489	.158	.293	.503	.729
1,000	.280	.639	.229	.584	.190	.345	.555	.814
2,000	.319	.696	.234	.595	.220	.380	.570	.855
5,000	.354	.756	.265	.646	.257	.401	.601	.878
10,000	.367	.770	.277	.685	.286	.416	.613	.890
20,000	.386	.799	.296	.724	.297	.423	.632	.898
from other languages to English								
Train Size	Hindi		Tamil		Japanese		Arabic	
	Exact	Fuzzy	Exact	Fuzzy	Exact	Fuzzy	Exact	Fuzzy
500	.237	.666	.123	.394	.111	.426	.214	.574
1,000	.266	.714	.148	.449	.122	.469	.250	.630
2,000	.276	.745	.150	.457	.134	.491	.265	.661
5,000	.295	.780	.159	.478	.144	.526	.280	.689
10,000	.308	.795	.160	.490	.154	.544	.284	.698
20,000	.311	.802	.170	.507	.166	.571	.289	.708

Figure 2: Transliteration among a Set of Languages

as most Japanese strings were concatenated strings (even when the corresponding English string is, say, *Sir Issac Newton*). We introduced a pre-processing module to force the alignment boundaries for Indic scripts to a pre-specified set (by observing the fact that no alignment boundaries can occur between a base character and a combining diacritic), which resulted in significant improvement in the transliteration accuracy. Though such additions are not generic, the workbench allows us to experiment with language-specific modules to boost transliteration accuracy on specific cases.

4. CONCLUSIONS

In this paper we outlined a generic framework for experimenting with transliteration tasks, in a modular manner. While our system displays a reasonable quality in transliteration, a shade below the state-of-the-art for specific language pairs, its strength lies in the fact that new languages may be added easily, and an effective transliteration mechanism may be developed quickly by experimentation with reusable components, a scripting language and language-specific modules. We plan to release this system as a workbench for pedagogical purposes to let users explore various algorithms and training processes in a systematic manner on a wide variety of languages. In addition, it may provide a generic transliteration system for resource-poor languages.

5. REFERENCES

[1] AbdulJaleel, N. and Larkey, L. Statistical transliteration for English-Arabic CLIR. *CIKM*, 2003.

[2] Al-Onaizan, Y. and Knight, K. Machine transliteration of names in Arabic text. *Comp. Approaches for Semitic Languages*, 2002.

[3] Brown, F. B. *et al.* The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 1993.

[4] Haizhou, L., Min, Z. and Jian, S. A joint source-channel model for machine transliteration. *42nd Meeting of ACL*, 2004.

[5] Virga, P. and Khudanpur, S. Transliteration of proper names in cross-lingual information retrieval. *Multiling. and Mixed-lang. Named Entity Recognition*, 2003.