Combining MDL Transliteration Training with Discriminative Modeling

Dmitry Zelenko
4300 Fair Lakes Ct.
Fairfax, VA 22033, USA
dmitry_zelenko@sra.com

Abstract

We present a transliteration system that introduces minimum description length training for transliteration and combines it with discriminative modeling. We apply the proposed approach to transliteration from English to 8 non-Latin scripts, with promising results.

1 Introduction

Recent research in transliteration and translation showed utility of increasing the n-gram size in transliteration models and phrase tables (Koehn et al., 2003). Yet most learning algorithms for training n-gram transliteration models place restrictions on the size of n-gram due to tractability and overfitting issues, and, in the case of machine translation, construct the phrase table after training the model, in an ad-hoc manner. In this paper, we present a minimum description length (MDL) approach (Grunwald, 2007) for learning transliteration models comprising n-grams of unrestricted size. Given a bilingual dictionary of transliterated data we seek to derive a transliteration model so that the combined size of the data and the model is minimized.

Use of discriminative modeling for transliteration and translation is another promising direction allowing incorporation of arbitrary features in the transliteration process (Zelenko and Aone, 2006; Goldwasser and Roth, 2008). Here we propose to use the transliteration model derived via MDL training as a starting point and learn the model weights in the discriminative manner. The discriminative approach also provides a natural way to integrate the language modeling component into the transliteration decoding process.

We experimentally evaluate the proposed approach on the standard datasets for the task of transliterating from English to 8 non-Latin scripts

2 MDL Training for Transliteration

In our transliteration setting, we are given a string e written in an alphabet V_1 (e.g., Latin), which is to be transliterated into a string f written in an alphabet V_2 (e.g., Chinese). We consider a transliteration process that is conducted by a transliteration model T, which represents a function mapping a pair of strings (e_i, f_i) into a score $T(e_i, f_i) \in R$. For an alignment [1] $A = \{(e_i, f_i)\}$ of e and f, we define the alignment score $T(A) = \sum_i T(e_i, f_i)$. For a string e and a model T, the decoding process seeks the optimal transliteration $T(e)$ with respect to the model T:

$$T(e) = arg\max_f \{\ T(A) \mid \exists A = \{(e_i, f_i)\}\ \}$$

Different assumptions for transliteration models lead to different estimation algorithms. A popular approach is to assume a *joint* generative model for pairs (e, f), so that given an alignment $A = \{(e_i, f_i)\}$, a probability $P(e, f)$ is defined to be $\prod_i p(e_i, f_i)$. The probabilities $p(e_i, f_i)$ are estimated using the EM algorithm, and the corresponding transliteration model is $T(e_i, f_i) = \log(p(e_i, f_i))$. We can alternatively model the *conditional* probability directly: $P(f|e) = \prod_i p(f_i|e_i)$, where we again estimate the conditional probabilities $p(f_i|e_i)$ via the EM algorithm, and define the transliteration model accordingly: $T(e_i, f_i) = \log(p(f_i|e_i))$. We can also combine joint estimation with conditional decoding, observing that $p(f_i|e_i) = \frac{p(e_i,f_i)}{\sum_f p(e_i,f_i)}$ and using the conditional transliteration model after estimating a joint generative model.

Increasing the maximum n-gram size in probabilistic modeling approaches, at some point, degrades model accuracy due to overfitting. Therefore, probabilistic approaches typically use a small n-gram size, and perform additional modeling *post*

[1] Here we consider only monotone alignments.

Proceedings of the 2009 Named Entities Workshop, ACL-IJCNLP 2009, pages 116–119,
Suntec, Singapore, 7 August 2009.

factum: examples include joint n-gram modeling and phrase table construction in machine translation.

We propose to apply the MDL principle to transliteration modeling by seeking the model that compresses the transliteration data so that the combined size of the compressed data and the model is minimized. If T corresponds to a joint probabilistic model $P = \{p(e_i, f_i)\}$, then we can use the model to encode the data $D = \{(e, f)\}$ in

$$\begin{aligned} C_D(P) &= -\sum_{(e,f)} \log P(e, f) \\ &= -\sum_{(e,f)} max_A \sum_i \log p(e_i, f_i) \end{aligned}$$

bits, where $A = \{(e_i, f_i)\}$ is an alignment of e and f.

We can encode each symbol of an alphabet V using $\log |V|$ bits so encoding a string s of length $|s|$ from alphabet V takes $C_V(s) = \log |V|(|s| + 1)$ bits (we add an extra string termination symbol for separability). Therefore, we encode each transliteration model in

$$C_T(P) = \sum_{(e_i,f_i)} C_T(e_i, f_i)$$

bits, where $C_T(e_i, f_i) = C_{V_1}(e_i) + C_{V_2}(f_i) - \log p(e_i, f_i)$ is the number of bits used to encode both the pair (e_i, f_i) and its code according to P. Thus, we seek a probability distribution P that minimizes $C(P) = C_D(P) + C_T(P)$.

Let P be an initial joint probability distribution for a transliteration model T such that a string pair (e_i, f_i) appeared $n(e_i, f_i)$ times, and $p(e_i, f_i) = n(e_i, f_i)/N$, where $N = \sum_{(e_i,f_i)} n(e_i, f_i)$. Then, encoding a pair (e_i, f_i) takes on average $C(e_i, f_i) = \frac{C_T(e_i,f_i)}{n(e_i,f_i)} - \log p(e_i, f_i)$ bits - here we distribute the model size component to all occurrences of (e_i, f_i) in the data. Notice that the combined data and model size $C(P) = \sum_{(e_i,f_i)} n(e_i, f_i)C(e_i, f_i)$. It is this quantity $C(e_i, f_i)$ that we propose to use when conducting the MDL training algorithm below.

1. Pick an initial P. Compute $C(e_i, f_i) = \frac{C_T(e_i,f_i)}{n(e_i,f_i)} - \log p(e_i, f_i)$. Set combined size $C(P) = \sum_{(e_i,f_i)} n(e_i, f_i)C(e_i, f_i)$.

2. Iterate: during each iteration, for each $(e, f) \in D$, find the minimum codesize alignment $A = arg\min_A \sum_i C(e_i, f_i)$ of (e, f). Use the alignments to re-estimate P and re-compute C. Exit when there is no improvement in the combined model and data size.

Experimentally, we observed fast convergence of the above algorithm just after a few iterations, though we cannot present a convergence proof as yet. We picked the initial model by computing co-occurrence counts of n-gram pairs in D, that is, $n(e_i, f_i) = \sum_{(e,f)} \min(n_e(e_i), n_f(f_i))$, where $n_e(e_i)$ ($n_f(f_i)$) is the number of times the n-gram e_i (f_i) appeared in the string e (f).

Note that a Bayesian interpretation of the proposed approach is not straightforward due to the use of empirical component $-\log p(e_i, f_i)$ in model encoding. Changing the model encoding to use, for example, a code for $n(e_i, f_i)$ would allow for a direct Bayesian interpretation of the proposed code, and we plan to pursue this direction in the future.

The output of the MDL training algorithm is the joint probability model P that we use to define the transliteration model weights as the logarithm of corresponding conditional probabilities: $T(e_i, f_i) = \log \frac{p(e_i,f_i)}{\sum_f p(e_i,f)}$. During the decoding process of inferring f from e via an alignment A, we integrate the language model probability $p(f)$ via a linear combination: $T_{GEN}(e) = arg\max_f\{T(A) + \mu \log p(f)/|f|\}$, where μ is a combination parameter estimated via cross-validation.

3 Discriminative Training

We use the MDL-trained transliteration model T as a starting point for discriminative training: we consider all n-gram pairs (e_i, f_i) with nonzero probabilities $p(e_i, f_i)$ as features of a linear discriminative model T_{DISCR}. We also integrate the normalized language modeling probability $p_0(f) = p(f)^{\frac{1}{|f|}}$ in the discriminative model as one of the features: $T_{DISCR}(e) = arg\max_f\{T(A) + T_0 p_0(f)\}$. We learn the weights $T(e_i, f_i)$ and T_0 of the discriminative model using the average perceptron algorithm of (Collins, 2002). Since both the transliteration model and the language model are required to be learned from the same data, and the language modeling probability is integrated into our decoding process, we remove the string e from the language model before processing the example (f, e) during

training; we re-incorporate the string e in the language model after the example (f, e) is processed by the averaged perceptron algorithm. We use the discriminatively trained model as the "standard" system in our experiments.

4 Experiments

We use the standard data for transliterating from English into 8 non-Latin scripts: Chinese (Haizhou et al., 2004); Korean, Japanese (Kanji), and Japanese (Katakana) (CJK Institute, 2009); Hindi, Tamil, Kannada, and Russian (Kumaran and Kellner, 2007). The data is provided as part of the Named Entities Workshop 2009 Machine Transliteration Shared Task (Li et al., 2009).

For all 8 datasets, we report scores on the standard tests sets provided as part of the evaluation. Details of the evaluation methodology are presented in (Li et al., 2009).

4.1 Preprocessing

We perform the same uniform processing of data: names are considered sequences of Unicode characters in their standard decomposed form (NFD). In particular, Korean Hangul characters are decomposed into Jamo syllabary. Since the evaluation data are provided in the re-composed form, we re-compose output of the transliteration system.

We split multi-word names (in Hindi, Tamil, and Kannada datasets) in single words and conducted training and evaluation on the single word level. We assume no word order change for multi-word names and ignore name pairs with different numbers of words.

4.2 System Parameters and Tuning

We apply pre-set system parameters with very little tuning. In particular, we utilize a 5-gram language model with Good-Turing discounting. The MDL training algorithm requires only the cardinalities of the corresponding alphabets as parameters, and we use the following approximate vocabulary sizes typically rounded to the closest power of 2 (except for Chinese and Japanese): for English, Russian, Tamil, and Kannada, we set $|V| = 32$; for Katakana and Hindi, $|V| = 64$; for Korean Jamo, $|V| = 128$; for Chinese and Japanese Kanji, $|V| = 1024$.

We perform 10 iterations of the average perceptron algorithm for discriminative training. For both alignment and decoding, we use a beam search decoder, with the beam size set to 100.

	Init	Comp	Ratio	Dict
Chinese	333 Kb	158 Kb	0.48	5780
Hindi	159 Kb	72 Kb	0.45	1956
Japanese (Kanji)	170 Kb	82 Kb	0.48	4394
Kannada	131 Kb	62 Kb	0.48	2010
Japanese (Katakana)	289 Kb	136 Kb	0.47	3383
Korean	69 Kb	31 Kb	0.45	1181
Russian	78 Kb	37 Kb	0.48	865
Tamil	134 Kb	62 Kb	0.46	1827

Table 1: MDL Data and Model Compression showing initial data size, final combined data and model size, the compression ratio, and the number of n-gram pairs in the final model.

	T1(Acc)	T2(Acc)	T2(F)	T2(MRR)
Chinese	0.522	0.619	0.847	0.711
Hindi	0.312	0.409	0.864	0.527
Japanese (Kanji)	0.484	0.509	0.675	0.6
Kannada	0.227	0.345	0.854	0.462
Japanese (Katakana)	0.318	0.420	0.807	0.541
Korean	0.339	0.413	0.702	0.524
Russian	0.488	0.566	0.919	0.662
Tamil	0.267	0.374	0.880	0.512

Table 2: Experimental results for transliteration from English to 8 non-Latin scripts comparing performance of generative (T1) and corresponding discriminative (T2) models.

4.3 Results

Our first set of experiments illustrates compression achieved by MDL training. Table 1 shows for each for the training datasets, the original size of the data, compressed size of the data including the model size, the compression ratio, and the number of n-gram pairs in the final model.

We see very similar compression for all languages. The number of n-gram pairs for the final model is also relatively small. In general, MDL training with discriminative modeling allows us to discover a flexible small set of features (n-gram pairs) without placing any restriction on n-gram size. We can interpret MDL training as search-

ing implicitly for the best bound on the n-gram size together with searching for appropriate features. Our preliminary experiments also indicate that performance of models produced by the MDL approach roughly corresponds to performance of models trained with the optimal bound on the size of n-gram features.

Table 2 demonstrates that discriminative modeling significantly improves performance of the corresponding generative models. In this setting, the MDL training step is effectively used for feature construction: its goal is to automatically hone in on a small set of features whose weights are later learned by discriminative methods.

From a broader perspective, it is an open question whether seeking a compact representation of sequential data leads to robust and best-performing models, especially in noisy environments. For example, state-of-the-art phrase translation models eschew succinct representations, and instead employ broad redundant sets of features (Koehn et al., 2003). On the other hand, recent research show that small translation models lead to superior alignment (Bodrumlu et al., 2009). Therefore, investigation of the trade-off between robust redundant and succinct representation present an interesting area for future research.

5 Related Work

There is plethora of work on transliteration covering both generative and discriminative models: (Knight and Graehl, 1997; Al-onaizan and Knight, 2002; Huang et al., 2004; Haizhou et al., 2004; Zelenko and Aone, 2006; Sherif and Kondrak, 2007; Goldwasser and Roth, 2008). Application of the minimum description length principle (Grunwald, 2007) in natural language processing has been heretofore mostly limited to morphological analysis (Goldsmith, 2001; Argamon et al., 2004). (Bodrumlu et al., 2009) present a related approach on optimizing the alignment dictionary size in machine translation.

6 Conclusions

We introduced a minimum description length approach for training transliteration models that allows to avoid overfitting without putting apriori constraints of the size of n-grams in transliteration models. We plan to apply the same paradigm to other sequence modeling tasks such as sequence classification and segmentation, in both supervised and unsupervised settings.

References

Y. Al-onaizan and K. Knight. 2002. Machine transliteration of names in arabic text. In *ACL Workshop on Comp. Approaches to Semitic Languages*, pages 34–46.

S. Argamon, N. Akiva, A. Amir, and O. Kapah. 2004. Efficient unsupervised recursive word segmentation using minimum description length. In *Proceedings of COLING*.

T. Bodrumlu, K. Knight, and S. Ravi. 2009. A new objective function for word alignment. In *Proceedings NAACL Workshop on Integer Linear Programming for NLP*.

CJK Institute. 2009. http://www.cjk.org.

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*.

J. Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, pages 153–198.

D. Goldwasser and D. Roth. 2008. Transliteration as constrained optimization. In *Proceedings of EMNLP*.

P. Grunwald. 2007. *The Minimum Description Length principle*. MIT Press.

L. Haizhou, Z. Min, and S. Jian. 2004. A joint source-channel model for machine transliteration. In *Proceedings of ACL*.

F. Huang, S. Vogel, , and A. Waibel. 2004. Improving named entity translation combining phonetic and semantic similarities. In *Proceedings of HLT/NAACL*.

K. Knight and J. Graehl. 1997. Machine transliteration. *Computational Linguistics*, pages 128–135.

P. Koehn, F. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NLT/NAACL*.

A. Kumaran and T. Kellner. 2007. A generic framework for machine transliteration. In *Proceedings of SIGIR*.

Haizhou Li, A. Kumaran, Min Zhang, and V. Pervouchine. 2009. Whitepaper of news 2009 machine transliteration shared task. In *Proceedings of ACL-IJCNLP 2009 Named Entities Workshop (NEWS 2009)*.

T. Sherif and G. Kondrak. 2007. Substring-based transliteration. In *Proceedings of ACL*.

D. Zelenko and C. Aone. 2006. Discriminative methods for transliteration. In *Proceedings of EMNLP*.