3-2004

Phoneme-based transliteration of foreign names for OOV problem

Wei GAO
Singapore Management University, weigao@smu.edu.sg

Kam-Fai WONG

Wai LAM

Phoneme-Based Transliteration of Foreign Names for OOV Problem

Wei Gao, Kam-Fai Wong, and Wai Lam

Department of Systems Engineering and Engineering Management,
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
{wgao,kfwong,wlam}@se.cuhk.edu.hk

Abstract. A proper noun dictionary is never complete rendering name translation from English to Chinese ineffective. One way to solve this problem is not to rely on a dictionary alone but to adopt automatic translation according to pronunciation similarities, i.e. to map phonemes comprising an English name to the phonetic representations of the corresponding Chinese name. This process is called transliteration. We present a statistical transliteration method. An efficient algorithm for aligning phoneme chunks is described. Unlike rule-based approaches, our method is data-driven. Compared to source-channel based statistical approaches, we adopt a direct transliteration model, i.e. the direction of probabilistic estimation conforms to the transliteration direction. We demonstrate comparable performance to source-channel based system.

1 Introduction

In cross language information retrieval (CLIR), query expressed in a source language is used to retrieve information represented in a target language. It involves keyword translation from the source to the target language and document translation in the opposite direction. Proper nouns, i.e. names of people, places, companies, etc., are by far the most frequent targets in queries. Contemporary dictionary-based translation techniques are ineffective as name dictionaries can never be comprehensive. New foreign names appear almost daily; and they become unregistered vocabulary in the dictionary. This brings about the classical Out-Of-Vocabulary (OOV) problem in lexicography. OOV names can worsen the performance of translation and retrieval.

Based on phonology, foreign name can usually be translated, or more appropriately transliterated into its target counterpart in terms of pronunciation similarities between them. Transliteration rules are practically mapping templates between the phonemes of source and target names. Existing rule bases are compiled manually. They are not easy to expand and are mostly non-universal, i.e. they are subjected to the interpretation of individual producers. In Mandarin of China mainland, for instance, the name of "Bin Laden" can be translated as /ben la deng[1]/ (本拉登), /bin la deng/ (宾拉登), /ben la dan/ (本拉丹) and /bin la dan/ (宾拉丹). In Taiwan's Mandarin, similar transliteration confusions exist as well: "Hussein" corresponds to /hai shan/ (海珊), /ha shan/ (哈珊) and /hu sheng/ (胡笙). Chinese dialects further render rule-based approach inefficient. Some popular names of celebrities initially transliterated

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

K.-Y. Su et al. (Eds.): IJCNLP 2004, LNAI 3248, pp. 110–119, 2005.

by Cantonese are somehow spreading in Mandarin: /lang na du/ (朗拿度), the Cantonese transliteration of "Ronaldo", often appears in Mandarin news media as equivalent to its Mandarin transliteration /luo na er duo/ (罗納尔多). Thus, rule-based approach has been undermined and an effective data-driven transliteration method is required.

In this paper, we present a statistical method for phoneme-based transliteration of foreign names from English to Chinese. Phonological transformation knowledge is acquired automatically by machine learning from existing source-target name pairs. Unlike source-channel based methods, it starts off with the direct estimation on transliteration model, which is then incorporated with target language model for the post-correction of generated hypotheses. Section 2 summarizes related work, in particular with source-channel model; Section 3 presents our model in detail; Section 4 elaborates the implementation of the model; Section 5 gives experimental results and analysis; Section 6 concludes the paper.

2 Related Work

Virga and Khudanpur [8] described a data-driven transliteration technique based on IBM's source-channel model, which was initially proposed for French-to-English statistical machine translation [3]. The fundamental equation is from Bayes' rule:

$$\hat{C} = \arg\max_{C} p(C \mid E) = \arg\max_{C} p(E \mid C)p(C) = \arg\max_{c_1 c_2 \ldots c_{|C|}} p(e_1^{|E|} \mid c_1^{|C|})\, p(c_1^{|C|}) \cdot \quad (1)$$

where $E = e_1^{|E|} = e_1 e_2 \ldots e_{|E|}$ denotes a $|E|$-phoneme English name as the observation on channel output, and $C = c_1^{|C|} = c_1 c_2 \ldots c_{|C|}$ represents E's $|C|$-phoneme Chinese translation as the source of channel input. $|E|$ and $|C|$ are the number of sound units they contain. The channel decoder reverses the direction, i.e. to find the most probable pinyin sequence $\hat{C}$ given an observation E, which indirectly maximizes the posterior probability $p(C|E)$ via optimal combination of the *inverted*-transliteration model $p(E|C)$ and the target language model $p(C)$. $p(E|C)$ was trained on name pairs represented by ARPABET symbols at English side and pinyin notations at Chinese side. It proceeded with a standard bootstrapping of IBM's translation model training in GIZA++ [1]: 5 EM iterations of Model-1 followed by 5 of Model-2, 10 of HMM and 10 of Model-4. $p(C)$ was trained on a pinyin vocabulary using tri-gram with Good-Turing smoothing and Katz back-off by CMU-Cambridge Language Modeling Toolkits [4]. Decoding was done by using USC-ISI ReWrite Decoder [5]. Note that the estimation of $p(E|C)$ is in the reversed direction, i.e. from Chinese to English. In fact, this is within the same framework as the generative model for Japanese-to-English backward transliteration proposed by [6]. The method demonstrated pinyin error rates in edit distance by 42.5%~50.8% on different data [8].

3 Our Forward Transliteration Model

3.1 Pitfalls of Source-Channel Model

The source-channel model in Eq-1 has two limitations for our task:

1. It is hard to extend the baseline of transliteration model by introducing additional dependencies [7], such as flexible neighboring or contextual phoneme features;
2. It allows only one target language phoneme to be associated with a contiguous group of source language phonemes, but not vice versa. The example in [8] exposes this limitation (see Fig. 1): /u/ and the second /i/ in the third line have to be looked as "spuriously" produced from dumb sound ε. Under IBM's model, such "inserted" symbols are known as *zero fertility* "words". They are "deleted" by source-channel during training and "reproduced" when decoding by considering inserting one of them before each target symbol of each remaining unaligned source phoneme according to the number of possible *zero fertility* symbols [5]. Although adding *zero fertility* symbols may increase the probability of hypotheses, incorrect transliterations are still abundant as such insertions are frequent. Without the loss of generality, it would be more natural and easier to handle if /f-u/ and /s-i/ were looked as initial-final clusters converted from single English phoneme /F/ and /S/. This is feasible under our direct model.

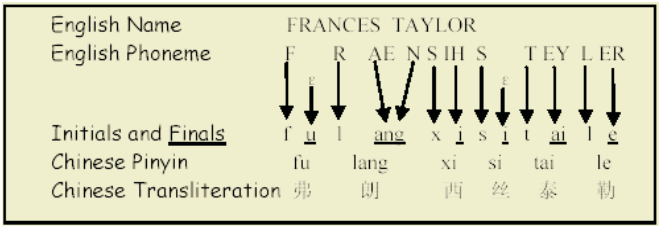

Fig. 1. An example depicting the process of English-to-Chinese transliteration in [8]

3.2 Soundness of Direct Model

We substitute $p(C|E)$ for $p(E|C)$ in Eq-1 so that we have a different forward transliteration method as follows:

$$\hat{C} = \arg\max_{C} p(C \mid E)\, p(C) = \arg\max_{c_1 c_2 \ldots c_{|C|}} p(c_1^{|C|} \mid e_1^{|E|})\, p(c_1^{|C|}) \cdot \tag{2}$$

The basic idea is that a direct transliteration model $p(C|E)$ concentrates on producing the most likely transcriptions for a given E, probably with ill-formed pinyin sequences though, and the language model $p(C)$ helps make corrections on the forms, e.g. eliminating illegal pinyin strings and yields better rankings of result syllables. Although Eq-2 is beyond the Bayes' theorem, it is mathematically sound under the more general Maximum Entropy (MaxEnt) principle [2,7]. We explain it briefly for completeness.

MaxEnt is a well-founded framework for directly modeling the posterior probability, where a set of M feature functions $h_m(E, C)$ and their corresponding model parameters λ_m, $m = 1, \ldots, M$, are introduced. The direct transliteration probability is given by:

$$p(C \mid E) = p_{\lambda_1^M}(C \mid E) = \frac{\exp\left[\sum_{m=1}^{M} \lambda_m h_m(E, C)\right]}{\sum_{E'} \exp\left[\sum_{m=1}^{M} \lambda_m h_m(E', C)\right]} \cdot \tag{3}$$

[7] suggested we could obtain the target sequence $\hat{C}$ that maximizes the posterior probability by omitting the normalization constant denominator:

$$\hat{C} = \arg\max_{C} p(C \mid E) = \arg\max_{C} \left\{ \exp\left[\sum_{m=1}^{M} \lambda_m h_m(E, C) \right] \right\}. \quad (4)$$

Then we select two feature functions and parameters: $h_1(E, C) = \log p_\theta(C|E)$, $h_2(E, C) = \log p_\gamma(C)$, $\lambda_1 = \lambda_2 = 1$. Thus the Eq-2 maximization obtains in combination of the direct $p_\theta(C|E)$ and $p_\gamma(C)$ with respect to model parameters θ and γ. The optimal parameters are to be estimated individually on parallel training corpus.

We summarize 4 possible general conditions mapping an English phoneme to items in pinyin's vocabulary in accordance with this direct model:

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

4 Direct Model Training

4.1 Alignment of Phoneme Chunks

We first introduce alignment indicators in a pair of sound sequences. Within total 39 phonemes (24 consonants, 15 vowels) in the English sound inventory and 58 pinyin symbols (23 initials and 35 finals) in Chinese, there are always some indicative sound units (indicators) that help for alignment. For *E*, they are: all the consonants; vowel at the first position; and the second vowel of two contiguous vowels. In C, accordingly, they are: all the initials; final at the first position; and the second final of two contiguous finals. Note that similar indicators are easily identifiable in other Romanized systems in Chinese. They are independent of alignment model.

We define the following variables: $\tau(S)$ = *# of indicators in sequence S,* $t = \max\{\tau(E), \tau(C)\}$, *and* $d = |\tau(E) - \tau(C)|$. We chunk *E* and *C* by tagging their indicators and compensate the one with fewer indicators by inserting *d* dumb sound ε(s) at its $\min\{\tau(E), \tau(C)\}$ possible positions ahead of its indicators. ε is practically also an indicator defined for alignment. This ensures that both sequences end up with the same number of indicators. The *t* chunks separated by indicators in *E* should align to the corresponding *t* chunks in *C* in the same order. They are called *alignment chunks*. There are $\| A \| = \left[\frac{P_t^d}{d!} \right] = \frac{t!}{(t-d)!\,d!}$ number of possible alignments at chunk level with respect to different positions of ε.

This method can guarantee each chunk contains two sound units at most. Thus, in a pair of aligned chunks, only three mapping layouts are possible for individual units:

1. e-to-c_1c_2: The alignment would be e-to-c_1c_2 where c_1c_2 is considered as an initial-final cluster;
2. e_1e_2-to-c_1c_2: The alignment would be e_1-to-c_1 and e_2-to-c_2;
3. e_1e_2-to-c: By adding a ε at *C* side, the alignment would be considered as e_1-to-c and e_2-to-ε or e_1-to-ε and e_2-to-c. In this case, we update $||A|| = ||A|| + 1$.

Fig. 2 shows the alignment chunks (indicators are tagged using '|') between the example pair /AE L B AH K ER K IY/ (Albuquerque) and /a er bo ke er ji/ (阿尔伯克尔基), where 5 alignments are possible at chunk level. But the total possible alignments would be 9 due to the existence of /K ER/ - /er/ in the first 4 alignments.

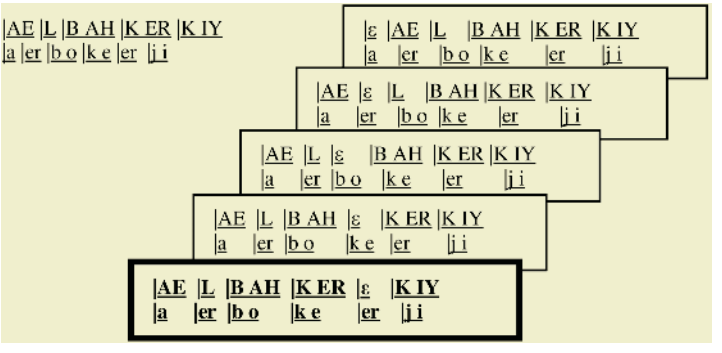

Fig. 2. An example depicting alignments of phoneme chunks between a name pair

4.2 EM Training for Symbol-Mapping Probabilities

We then applied EM training to find the most probable alignment (Viterbi alignment) for each name pair and compute symbol-mapping probabilities. The training goes as follows:

1) *Initialization*: For each name pair, assign equal weights $||A||^{-1}$ to all alignments based on phoneme chunks.
2) *Expectation*: For each of the 40 English phonemes, count the instances of its different mappings on all alignments produced. Each alignment contributes counts in proportion to its own weight. Normalize the scores of the pinyin sound units it maps to so that the mapping probability sums to 1.
3) *Maximization*: Re-compute the alignment scores. Each alignment is scored with the product of the scores of the symbol-mappings it contains. Normalize the alignment scores so that each pair's alignments scores sum to 1.
4) *Repeat* step 2-3 until the symbol-mapping probabilities converge, i.e. the variation of each probability between two iterations becomes less than a specified threshold.

Compared to the brutal-force alignment [6], our EM training based on aligned phoneme chunks produces much fewer possible alignments, thus fewer possible mappings for each English phoneme. Mappings crossing chunks are also avoided. Therefore, the symbol-mappings tend to be more accurate. For each pair, the Viterbi alignment is found whose alignment score (weight) approaches to 1 with the iteration.

Dumb sound ε is introduced to both sides of phonetic alphabets during the processing of phoneme chunks. It plays an important role for the case 3 and 4 in Section 3.2. The EM training also calculates the transition probabilities from English phonemes to initial-final clusters. The algorithm identifies these clusters and dynamically appends them to pinyin inventory as additional candidates for transcriptions. This can improve the shortcomings caused by *zero fertility* symbols in source-channel model.

4.3 WFST for Phonetic Transition

We then build a weighted finite state transducer (WFST) based on the symbol-mapping table for the transcription of each English phoneme into its possible pinyin counterparts. Each arc carries the transition labels and transition costs. Fig. 3 shows part of the transducer. Note that the arcs such as [AA:uo|0.904] are split into multiple arcs, i.e. [AA:u|0.904] and [ε:o|0] jointed by intermediate nodes 1, 2,...5,... This is for the following transducer for pinyin syllable segmentation being able to connect with it.

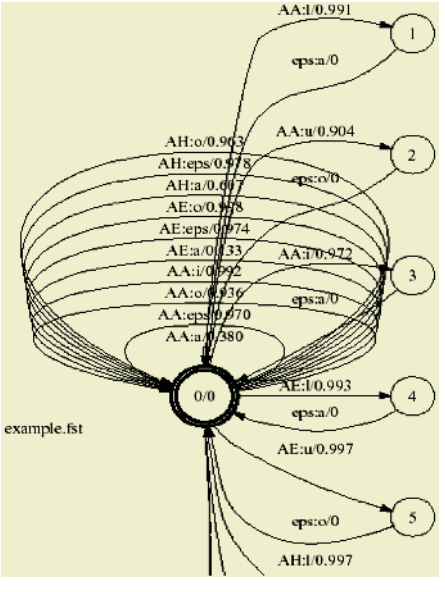

Fig. 3. Part of the WFST[2] based on *p(C|E)*

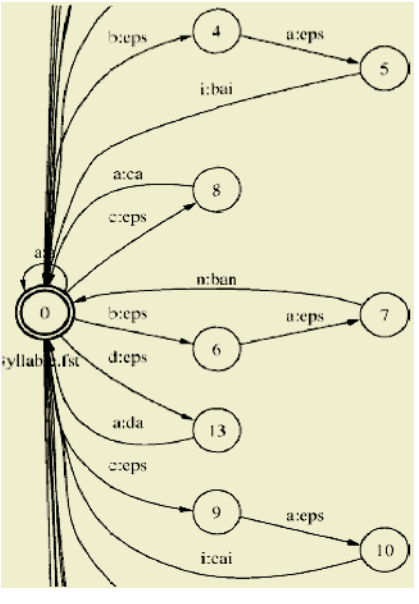

Fig. 4. Part of the FST for pinyin syllabification

4.4 Issues of Illegal Pinyin Syllables

Many pinyin symbol sequences produced by the transliteration model cannot be syllabified or include illegitimate syllables as the transducer has no knowledge about pinyin's regulations. Actually only 396 of 23*25 possible combinations of initials and finials can constitute legal pinyin syllables. We can easily collect them from our cor-

[2] Label "eps" in the figure represents dumb sound ε (epsilon).

pus by using an automatic scanning program. Based on this knowledge, we construct a FST as shown in Fig. 4. It is composed with the previous WFST for sifting out illegal pinyin sequences and segmenting them into syllables.

4.5 Language Model Training and Bi-gram WFSA

A syllable-based bi-gram language model of pinyin is trained using the Chinese part of the same 41,674 instances, on which the transliteration WFST is built. The model *P(C)* is approximated by counting the frequencies of syllable occurrences in this data set using the equation:

$$p(C) \cong \prod_i p(c_i \mid c_{i-1}) \cong \prod_i \frac{count\ (c_i c_{i-1})}{count\ (c_i)} \ . \tag{5}$$

where c_i is the pinyin syllable of a Chinese character.

We then implement the bi-gram model using a weighted finite state acceptor (WFSA) with one state for each item in the pinyin syllable vocabulary. Between each pair of states, say *x*/*y*, there is a single transition whose label is the syllable *y* and whose probability is *p(y|x)*. We then add a special final state with transitions leading to it from every other state labeled by ε with probability 1.0. Finally, a start state is added with transitions to every state *y* with label *y* and probability *p(y)*.

The WFSA is used to re-rank the pinyin syllable sequences yielded from the composition of the previous two transducers and the search results of *k*-best path algorithm. Because the search space of the syllables produced by the previous transducers is extremely large, we apply the bi-gram search to only the first few hundred candidates for each English name.

5 Experiments and Evaluation

5.1 Similarity Measurement

Similarity measurement is based on edit distance, which is defined as the minimum number of insertions, deletions and substitutions required for transforming one string to the other. The performance of transliteration is measured by error rate, which is defined as: $e = d(S_1, S_2) / |S_2|$, where S_1 is the machine transliteration and S_2 the standard, $d(S_1, S_2)$ denotes the edit distance between the two transliterations, and $|S_2|$ is the length of S_2.

The process is participated by a bilingual dictionary (LDC's English-Chinese bilingual named entity list beta v.1.0), an English pronunciation dictionary (CMU's pronunciation dictionary) and a Chinese character-pinyin table (LDC's Chinese character table with pinyin). We harvest 46,305 name pairs from the bilingual dictionary, all of which also appear in the pronunciation dictionary with deterministic phonemic representations. We obtain English name pronunciations and their Chinese equivalents by looking up the pronunciation dictionary and the character-pinyin conversion table. In the experiments, 41,674 name pairs are used for training, in which a portion of 4,631 pairs is used for close test, and the remaining 4,631 pairs for open test.

5.2 Experimental Results

Experiment I. Only top-1 machine transliteration of each name is chosen for comparison with the standard translation. We set up six error rate ranges: [0%], (0%~20%], (20%~40%], (40%~60%], (60%~80%] and (80%~100%][3]. We count the number of names whose error rate falls in each range and the accumulative percentages are listed in Table 1.

Table 1. Results of experiment I

Error rate (%)	0	0~20	20~40	40~60	60~80	80~100
Close	12.33%	10.39%	34.36%	28.34%	9.91%	4.67%
Open	10.20%	12.13%	33.58%	29.29%	10.40%	4.40%

Experiment II. The system yields top-50 candidates for each foreign name. We count the number of correct transliterations whose error rate is 0. The accumulative percentage is listed in Table 2. This test evaluates the proportion of instances whose correct transliteration can be found in top-n generated candidates.

Table 2. Results of experiment II

Top n	1	10	20	30	40	50
Close	12.33%	50.32%	59.20%	61.81%	62.98%	63.63%
Open	10.20%	46.56%	54.87%	57.82%	58.84%	59.23%

Experiment III. For comparison with source-channel based system, we implement the work desribed in [8]. We replicate their first traslation system [8] with the only exception that we obtain phoneme sequences of English names via looking up pronunciation dictionary instead of text-to-speech system. We test our system and our implementation of [8]'s system on the aforementioned data set. The averaged error rate of top-1 machine transliterations is shown in Table 3.

Table 3. Transliteration error rates compared to the source-channel based system

Systems	Source-Channel	Ours
Close	33.65%	38.00%
Open	34.85%	38.50%

So far, the transliteration model and the language model are both trained on the same 41,674 instances. To explore the influence of the two components separately, we train our transliteration model on the 4,631 instances for close test and keep the language model intact. We achieve error rate of 42.67% in open test. Then we train the language model on this 4,631 instances and reuse the previous transliteration model. This time, the error rate rises up to 46.78%.

[3] (M%~N%] denotes > M% AND ≤ N%.

5.3 Discussions

In experiment I, the possibility of finding correct transliterations in top-1 result candidates is fairly low, evidenced as only 12.33% and 10.20% of test instances end up with correct transliterations. If we consider acceptable transliterations whose error rate is less than 20%, the accumulative percentage would be 22.72% and 22.33% for the close and open test respectively. Two pinyin sequences having 20% edit distance can be exemplified as /ben la deng/ (本拉登) and /ben la dan/ (本拉丹). Hence, machine transliterations with 20% error rate or less can be considered as phonetically equivalent but misspelled.

From experiment II where top-50 transliterations are examined, nearly half of the test instances (50.32% for close test and 46.56% for open test) can have their correct transliterations within top-10 transliteration candidates. We also note the considerable increase on the percentage of correct transliterations if we examine top-20 candidates compared to only top-1. But no apparent improvement is achieved if we further yield more top candidates.

In experiment III, our approach demonstrates comparable performance to source-channel based system, but slightly worse by 4.35% on close test and 3.65% on open test in averaged error rates. The error rates (between 30% and 40%) indicate that the top-1 transliterations from the two systems are generally acceptable. Error rate more than 50% should be unacceptable since two pinyin sequences with 50% difference in edit distance are identified as phonetic representations of nearly different names like /ya li shan da/ (亚力山大) and /ya li shi duo de/ (亚里士多德).

Furthermore, the distinct rise of error to 46.78% indicates that our approach is more sensitive to the language model than to the transliteration model. The paucity of data can affect the search using language model, evidenced by the serious error increase as the language model is trained on sparse data. The reason is that transliteration model only reflects symbol-mapping relationships among phonemes rather than sequences, leaving lots of work to be done by language model. The problem is supposed to be alleviated by improving the transliteration model further.

6 Conclusion and Future Work

We model the statistical transliteration problem as a direct phonetic symbol transcription model plus a language model for post-adjustment. The baseline indicates a comparable performance suggested by source-channel based system. The advantage of direct method is its flexibility for incorporating features with respect to dependencies among surrounding phonemes. We can expand our transliteration model in the future using contextual feature functions within MaxEnt framework [2]. Also, we will improve our language model using tri-gram for a better accuracy or smoothing techniques for overcoming data sparseness.

References

1. Al-Onaizan, Y., Curin, J., Jahr, M., Knight, K., Lafferty, J., Melamed, D., Och, F.J., Purdy, D., Smith, N.A., Yarowsky, D.: Statistical machine translation. Final Report of JHU Workshop (1999).

2. Berger, A.L., Della Pietra, S.A., Della Pietra, V.J.: A maximum entropy approach to natural language processing. Computational Linguistics (1996) 22(1): 39-72.
3. Brown, P.F., Della Pietra, S.A., Della Pietra, V.J., Mercer, R.L.: The mathematics of statistical machine translation: Parameter estimation. Computational Linguistics (1993) 19(2): 263-311.
4. Clarkson, P., Rosenfeld, R.: Statistical language modeling using the CMU-Cambridge toolkit. In Proc. of the 5th European Conf. on Speech Communication and Technology (1997) 2707-2710.
5. Germann, U., Jahr, M, Knight, K., Marcu, D., Yamada, K.: Fast decoding and optimal decoding for machine translation. In Proc. of the 39th Annual Meeting of ACL (2001) 228-235.
6. Knight, K., Graehl, J.: Machine transliteration. In Proc. of the 35th Annual Meeting of ACL (1997) 128-135.
7. Och, F.J., Ney, H.: Discriminative training and maximum entropy models for statistical machine translation. In Proc. of the 40th Annual Meeting of ACL (2002) 295-302.
8. Virga, P., Khudanpur, S.: Transliteration of proper names in cross-lingual information retrieval. In Proc. of the ACL Workshop on Multi-lingual Named Entity Recognition (2003).