A Phrase-Based Context-Dependent Joint Probability Model for Named Entity Translation

Min Zhang[1], Haizhou Li[1], Jian Su[1], and Hendra Setiawan[1,2]

[1] Institute for Infocomm Research,
21 Heng Mui Keng Terrace, Singapore 119613
{mzhang, hli, sujian, stuhs}@i2r.a-star.edu.sg
[2] Department of Computer Science,
National University of Singapore, Singapore, 117543
hendrase@comp.nus.edu.sg

Abstract. We propose a phrase-based context-dependent joint probability model for Named Entity (NE) translation. Our proposed model consists of a lexical mapping model and a permutation model. Target phrases are generated by the context-dependent lexical mapping model, and word reordering is performed by the permutation model at the phrase level. We also present a two-step search to decode the best result from the models. Our proposed model is evaluated on the LDC Chinese-English NE translation corpus. The experiment results show that our proposed model is high effective for NE translation.

1 Introduction

A Named Entity (NE) is essentially a proper noun phrase. Automatic NE translation is an indispensable component of cross-lingual applications such as machine translation and cross-lingual information retrieval and extraction.

NE is translated by a combination of meaning translation and/or phoneme transliteration [1]. NE transliteration has been given much attention in the literature. Many attempts, including phoneme and grapheme-based methods, various machine learning and rule-based algorithms [2,3] and Joint Source-Channel Model (JSCM) [4], have been made recently to tackle the issue of NE transliteration. However, only a few works have been reported in NE translation. Chen et al. [1] proposed a frequency-based approach to learn formulation and transformation rules for multilingual Named Entities (NEs). Al-Onaizan and Knight [5] investigated the translation of Arabic NEs to English using monolingual and bilingual resources. Huang et al. [6] described an approach to translate rarely occurring NEs by combining phonetic and semantic similarities. In this paper, we pay special attention to the issue of NE translation.

Although NE translation is less sophisticated than machine translation (MT) in general, to some extent, the issues in NE translation are similar to those in MT. Its challenges lie in not only the ambiguity in lexical mapping such as <副 (*Fu*),Deputy> and <副 (*Fu*),Vice> in Fig.1 in the next page, but also the position permutation and fertility of words. Fig.1 illustrates two excerpts of NE translation from the LDC corpus [7]:

R. Dale et al. (Eds.): IJCNLP 2005, LNAI 3651, pp. 600 – 611, 2005.

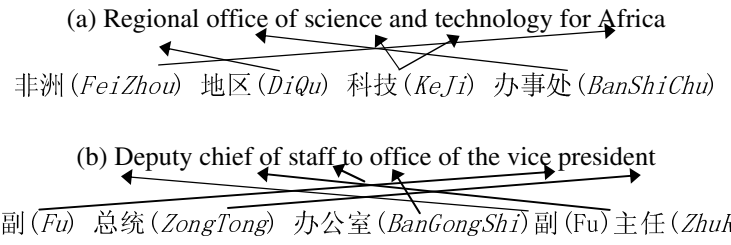

Fig. 1. Example bitexts with alignment

where the *italic* word is the Chinese *pinyin* transcription.

Inspired by the JSCM model for NE transliteration [4] and the success of statistical phrase-based MT research [8-12], in this paper we propose a phrase-based context-dependent joint probability model for NE translation. It decomposes the NE translation problem into two cascaded steps:

1) Lexical mapping step, using the phrase-based context-dependent joint probability model, where the appropriate lexical item in the target language is chosen for each lexical item in the source language;
2) Reordering step, using the phrase-based *n*-gram permutation model, where the chosen lexical items are re-arranged in a meaningful and grammatical order of target language.

A two-step decoding algorithm is also presented to allow for effective search of the best result in each of the steps.

The layout of the paper is as follows. Section 2 introduces the proposed model. In Section 3 and 4, the training and decoding algorithms are discussed. Section 5 reports the experimental results. In Section 6, we compare our model with the other relevant existing models. Finally, we conclude the study in Section 7.

2 The Proposed Model

We present our method by starting with a definition of translation unit in Section 2.1, followed by the formulation of the lexical mapping model and the permutation model in Section 2.2.

2.1 Defining Translation Unit

Phrase level translation models in statistical MT have demonstrated significant improvement in translation quality by addressing the problem of local re-ordering across language boundaries [8-12]. Thus we also adopt the same concept of phrase used in statistical phrase-based MT [9,11,12] as the basic NE translation unit to address the problems of word fertility and local re-ordering within phrase.

Suppose that we have Chinese as the source language $c_1^J = c_1 ... c_j ... c_J$ and English as the target language $e_1^I = e_1 ... e_i ... e_I$ in an NE translation (c_1^J, e_1^I), where

$c_j \in c_1^J$ and $e_i \in e_1^I$ are Chinese and English words respectively. Given a directed word alignment $\mathcal{A}$:{ $c_1^J \to e_1^I, e_1^I \to c_1^J$ }, the set of the bilingual phrase pairs Λ is defined as follows:

$$\begin{aligned}\Lambda(c_1^J, e_1^I, \mathcal{A}) = \{\ & (c_{j_1}^{j_2}, e_{i_1}^{i_2}) : \\ & \forall j \in \{j_1 \ldots j_2\}, \exists i \in \{i_1 \ldots i_2\} : j \to i \in \mathcal{A} \\ & \wedge \textit{vice versa}\ \}\end{aligned} \tag{1}$$

The above definition means that two phrases are considered to be translations of each other, if the words are aligned exclusively within the phrase pair, and not to the words outside [9,11,12]. The phrases have to be contiguous and a *null* phrase is not allowed.

Suppose that the NE pair (c_1^J, e_1^I) is segmented into *X* phrase pairs ($\tilde{c}_1^X, \tilde{e}_1^X$) according to the phrase pair set Λ, where $\tilde{e}_1^X$ is reordered so that the phrase alignment is in monotone order, *i.e.*, $\tilde{c}_x$ is aligned $\tilde{c}_x \leftrightarrow \tilde{e}_x$ For simplicity, we denote by $\lambda_x = <\tilde{c}_x, \tilde{e}_x>$ the x^{th} phrase pair in ($\tilde{c}_1^X, \tilde{e}_1^X$) $= \lambda_1 \ldots \lambda_x \ldots \lambda_X$, $\lambda_x \in \Lambda$.

2.2 Lexical Mapping Model and Permutation Model

Given the phrase pair set Λ, an NE pair (c_1^J, e_1^I) can be rewritten as ($\tilde{c}_1^X, \tilde{e}_1^X$) = $\lambda_1 \ldots \lambda_x \ldots \lambda_X = \lambda_1^X$. Let us describe a Chinese to English (C2E) bilingual training corpus as the output of a generative stochastic process:

---

(1) Initialize queue Q_c and Q_e as empty sequences;
(2) Select a phrase pair $\lambda_x = <\tilde{c}_x, \tilde{e}_x>$ according to the probability distribution $p(\lambda_x \mid \lambda_1^{x-1})$, remove λ_x from Λ;
(3) Append the phrase $\tilde{c}_x$ to Q_c and append the phrase $\tilde{e}_x$ to Q_e;
(4) Repeat steps 2) and 3) until Λ is empty;
(5) Reorder all phrases in Q_e according to the probability distribution of the permutation model;
(6) Output Q_e and Q_c.

As $p(\lambda_x \mid \lambda_1^{x-1})$ is typically obtained from a source-ordered aligned bilingual corpus, reordering is needed only for the target language. According to this generative story, the joint probability of the NE pair (c_1^J, e_1^I) can then be obtained by summing the probabilities over all possible ways of generating various sets of Λ and all possible permutations that can arrive at (c_1^J, e_1^I). This joint probability can be formulated

in *Eq.*(2). Here we assume that the generation of the set Λ and the reordering process are modeled by *n*-order Markov models, and the reordering process is independent of the source word position.

$$\begin{aligned} p(c_1^J, e_1^I) &= \sum_{\Lambda} \{ p(\lambda_1^X) * p(e_1^I \mid \tilde{e}_1^X) \} \\ &\approx \sum_{\Lambda} \{ (\prod_{x=1}^{X} p(\lambda_x \mid \lambda_{x-n}^{x-1})) * p(\tilde{e}_{k_1}^{k_X} \mid \tilde{e}_1^X) \} \end{aligned} \tag{2}$$

$$p(\tilde{e}_{k_1}^{k_X} \mid \tilde{e}_1^X) \approx \prod_{x=1}^{X} p(\tilde{e}_{k_x} \mid \tilde{e}_{k_{x-n}}^{k_{x-1}}) \tag{3}$$

where $\tilde{e}_{k_1}^{k_X}$ stands for one of the permutational sequences of $\tilde{e}_1^X$ that can yield e_1^I by linearly joining all phrases, i.e., $e_1^I = \tilde{e}_{k_1}^{k_X}$ (). The generative process, as formulated above, does not try to capture how the source NE is mapped into the target NE, but rather how the source and target translation units can be generated simultaneously in the source order and how the target NE can be constructed by reordering the target phrases, $\tilde{e}_1^X$.

In essence, our proposed model consists of two sub-models: a **lexical mapping model (LMM)**, characterized by $p(\lambda_x \mid \lambda_{x-n}^{x-1})$, that models the monotonic generative process of phrase pairs; and **a permutation model (PM)**, characterized by $p(\tilde{e}_{k_x} \mid \tilde{e}_{k_{x-n}}^{k_{x-1}})$, that models the permutation process for reordering of the target language. The **LMM** in this paper is among the first attempts to introduce context-dependent lexical mapping into statistical MT (Och *et al.*, 2003). The **PM** here is also different from the widely used position-based distortion model in that it models phrase connectivity instead of position distortion. Although **PM** functions as an *n*-gram language model, it only models the ordering connectivity between target language phrases, i.e., it is not in charge of target word selection.

Since the proposed model is phrase-based and we use conditional joint probability in **LMM** and use context-dependent *n*-gram in **PM**, we call the proposed model a phrase-based context-dependent joint probability model.

3 Training

Following the modeling strategy discussed above, the training process consists of three steps: phrase alignment, reordering of corpus, and learning statistical parameters for lexical mapping and permutation models.

3.1 Acquiring Phrase Pairs

To reduce vocabulary size and avoid sparseness, we constrain the phrase length to up to three words and the lower-frequency phrase pairs are pruned out for accurate

phrase-alignment[1]. Given a word alignment corpus which can be obtained by means of the publicly available GIZA++ toolkit [15], it is very straightforward to construct the phrase-alignment corpus by incrementally traversing the word-aligned NE from left to right[2]. The set of resulting phrase pairs forms a lexical mapping table.

3.2 Reordering Corpus

The context-dependent lexical mapping model assumes monotonic alignment in the bilingual training corpus. Thus, the phrase aligned corpus needs to be reordered so that it is in either source-ordered or target-ordered alignment. We choose to reorder the target phrases to follow the source order. Only in this way can we use the lexical mapping model to describe the monotonic generative process and leave the reordering of target translation units to the permutation model.

3.3 Training LMM and PM

According to *Eq.* (2), the **lexical mapping model (LMM)** and the **permutation model (PM)** can be interpreted as a kind of *n*-gram Markov model. The phrase pair is the basic token of **LMM** and the target phrase is the basic token of **PM**. A bilingual corpus aligned in the source language order is used to train **LMM**, and a target language corpus with phrase segmentation in their original word order is used to train **PM**. Given the two corpora, we use the SRILM Toolkit [13] to train the two *n*-gram models.

4 Decoding

The proposed modeling framework allows **LMM** and **PM** decoding to cascade as in Fig.2.

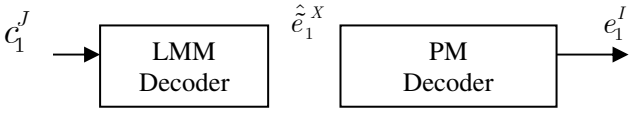

Fig. 2. A cascaded decoding strategy

The two-step operation is formulated by *Eq.*(4) and *Eq.*(5). Here, the probability summation as in *Eq.*(2) is replaced with maximization to reduce the computational complexity:

$$\hat{\tilde{e}}_1^X = \arg\max_{\Lambda}\{\prod_{x=1}^{X} p(\lambda_x \mid \lambda_{x-n}^{x-1})\} \tag{4}$$

[1] Koehn et. al. [12] found that that in MT learning phrases longer than three words and learning phrases from high-accuracy word-alignment does not have strong impact on performance.

[2] For the details of the algorithm to acquire phrase alignment from word alignment, please refer to the section 2.2 & 3.2 in [9] and the section 3.1 in [12].

$$\hat{e}_1^I = \arg\max_{\Omega}\{\prod_{x=1}^{X} p(\tilde{e}_{k_x} \mid \tilde{e}_{k_{x-n}}^{k_{x-1}})\} \quad (5)$$

LMM decoding: Given the input c_1^J, the **LMM** decoder searches for the most probable phrase pair set Λ in the source order using *Eq.*(4). Since this is a monotone search problem, we use a stack decoder [14,18] to arrive at the *n*-best results.

PM decoding: Given the translation phrase sequence $\hat{\tilde{e}}_1^X$ from the **LMM** decoder, the **PM** decoder searches for the best phrase order that gives the highest *n*-gram score by using *Eq.*(5) in the search space Ω, which is all the $X!$ permutations of the all phrases in $\hat{\tilde{e}}_1^X$. This is a non-monotone search problem.

The **PM** decoder conducts a time-synchronized search from left to right, where time clocking is synchronized over the number of phrases covered by the current partial path. To reduce the search space, we prune the partial paths along the way. Two partial paths are considered identical if they satisfy the following both conditions:

1) They cover the same set of phrases regardless of the phrase order;
2) The last *n*-1 phrases and their ordering are identical, where *n* is the order of the *n*-gram permutation model.

For any two identical partial paths, only the path with higher *n*-gram score is retained. According to *Eq.* (5), the above pruning strategy is risk-free because the two partial paths cover the exact same portion of input phrases and the *n*-gram histories for the next input phrases in the two partial paths are also identical.

It is also noteworthy that the decoder only needs to perform $X/2$ expansions as after $X/2$ expansions, all combinations of $X/2$ phrases would have been explored already. Therefore, after $X/2$ expansions, we only need to combine the corresponding two partial paths to make up the entire input phrases, then select the path with highest *n*-gram score as the best translation output.

Let us examine the number of paths that the **PM** decoder has to traverse. The pruning reduces the search space by a factor of $Z!$, from $P_X^Z = \frac{X!}{(X-Z)!}$ to $C_X^Z = \frac{X!}{Z! \bullet (X-Z)!}$, where Z is the number of phrases in a partial path. Since $C_X^Z = C_X^{X-Z}$, the maximum number of paths that we have to traverse is $C_X^{X/2}$. For instance, when $X = 10$, the permutation decoder traverses $C_{10}^5 = 252$ paths instead of the $P_{10}^5 = 30,240$ in an exhausted search.

By cascading the translation and permutation steps, we greatly reduce the search space. In **LMM** decoding, the traditional stack decoder for monotone search is very fast. In **PM** decoding, since most of NE is less than 10 phrases, the permutation decoder only needs to explore at most $C_{10}^5 = 252$ living paths due to our risk-free pruning strategy.

5 Experiments

5.1 Experimental Setting and Modeling

All the experiments are conducted on the LDC Chinese-English NE translation corpus [7]. The LDC corpus consists of a large number of Chinese-Latin language NE entries. Table 1 reports the statistics of the entire corpus. Because person and place names in this corpus are translated via transliteration, we only extract the categories of organization, industry, press, international organization, and others to form a corpus subset for our NE translation experiment, as indicated in bold in Table 1. As the corpus is in its beta release, there are still many undesired entries in it. We performed a quick proofreading to correct some errors and remove the following types of entries:

1) The duplicate entry;
2) The entry of single Chinese or English word;
3) The entries whose English translation contains two or more non-English words.

We also segment the Chinese translation into a word sequence. Finally, we obtain a corpus of 74,606 unique bilingual entries, which are randomly partitioned into 10 equal parts for 10-fold cross validation.

Table 1. Statistics of the LDC Corpus

Category	# of Entries	
	C2E	E2C
Person	486,212	572,213
Place	276,382	298,993
Who-is-Who	30,028	36,881
Organization	**30,800**	**37,145**
Industry	**54,747**	**58,468**
Press	**29,757**	**32,922**
Int'l Org	**7,040**	**7,040**
Others	**13,007**	**14,066**

As indicated in Section 1, although MT is more difficult than NE translation, they both have many properties in common, such as lexical mapping ambiguity and permutation/distortion. Therefore, to establish a comparison, we use the publicly available statistical MT training and decoding tools, which can represent the state-of-the-art of statistical phrase-based MT research, to carry out the same NE translation experiments as reference cases. All the experiments conducted in this paper are listed as follow:

1) IBM method C: word-based IBM Model 4 trained by GIZA++[3] [15] and ISI Decoder[4] [14,16];

[3] http://www.fjoch.com/
[4] http://www.isi.edu/natural-language/software/decoder/manual.html

2) IBM method D: phrase-based IBM Model 4 trained by GIZA++ on phrase-aligned corpus and ISI Decoder working on phrase-segmented testing corpus.
3) Koehn method: Koehn et al.'s phrase-based model [12] and PHARAOH[5] decoder[6];
4) Our method: phrase-based bi-gram **LMM** and bi-gram **PM**, and our two-step decoder.

To make an accurate comparison, all the above three phrase-based models are trained on the same phrase-segmented and aligned corpus, and tested on the same phrase-segmented corpus. ISI Decoder carries out a greedy search, and PHARAOH is a beam-search stack decoder. To optimize their performances, the two decoders are allowed to do unlimited reordering without penalty. We train trigram language models in the first three experiments and bi-gram models in the forth experiment.

5.2 NE Translation

Table 2 and Table 3 report the performance of the four methods on the LDC NE translation corpus. The results are interpreted in different scoring measures, which allow us to compare the performances from different viewpoints.

- *ACC* reports the accuracy of the exact;
- *WER* reports the word error rate;
- *PER* is the position-independent, or "bag-of-words" word error rate;
- *BLEU* score measures *n*-gram precision [19]
- *NIST* score [20] is a weighted *n*-gram precision.

Please note that *WER* and *PER* are error rates, the lower numbers represent better results. For others, the higher numbers represents the better results.

Table 2. E2C NE translation performance (%)

			IBM method C	IBM method D	Koehn method	Our method
E2C	Open test	*ACC*	**24.5**	**36.3**	**47.1**	**51.5**
		WER	51.0	38.5	32.5	26.6
		PER	**48.5**	**36.2**	**26.8**	**16.3**
		BLEU	**29.9**	**41.8**	**51.2**	**56.1**
		NIST	7.2	8.6	9.3	10.2
	Closed test	*ACC*	51.1	78.9	88.2	90.9
		WER	34.1	12.8	6.3	4.3
		PER	31.5	9.5	4.1	2.7
		BLEU	54.7	80.9	89.1	91.9
		NIST	11.1	14.2	14.7	14.8

[5] http://www.isi.edu/licensed-sw/pharaoh/
[6] http://www.isi.edu/licensed-sw/pharaoh/manual-v1.2.ps

Table 3. C2E NE translation performance (%)

			IBM method C	IBM method D	Koehn method	Our method
C2E	open test	*ACC*	**13.4**	**21.8**	**31.2**	**36.1**
		WER	60.8	45.8	41.3	38.9
		PER	**49.6**	**38.2**	**32.6**	**26.6**
		BLEU	**25.1**	**49.8**	**52.9**	**54.1**
		NIST	5.94	8.21	8.91	9.25
	closed test	*ACC*	34.3	69.5	79.2	81.3
		WER	48.2	23.6	11.3	9.2
		PER	35.7	14.7	8.7	6.2
		BLEU	42.5	76.2	85.7	88.0
		NIST	8.7	12.7	13.8	14.4

Table 2 & 3 show that our method outperforms the other three methods consistently in all cases and by all scores. IBM method D gives better performance than IBM method C, simply because it uses phrase as the translation unit instead of single word. Koehn et al.'s phrase-based model [12] and IBM phrase-based Model 4 used in IBM method D are very similar in modeling. They both use context-independent lexical mapping model, distortion model and trigram target language model. The reason why Koehn method outperforms IBM method D may be due to the different decoding strategy. However, we still need further investigation to understand why Koehn method outperforms IBM method D significantly. It may also be due to the different LM training toolkits used in the two experiments.

Our method tops the performance among the four experiments. The significant position-independent word error rate (*PER*) reduction shows that our context-dependent joint probability lexical mapping model is quite effective in target word selection compared with the other context-free conditional probability lexical model together with target word *n*-gram language model.

Table 4. Step by step top-1 performance (%)

	LMM decoder	**LMM+PM** decoder
E2C	59.9	51.5
C2E	40.5	36.1

Table 4 studies the performance of the decoder by steps. The **LMM** decoder column reports the top-1 "bag-of-words" accuracy of the **LMM** decoder regardless of word order. This is the upper bound of accuracy that the PM decoder can achieve. The **LMM+PM** decoder column shows the combined performance of two steps, where we

measure the top-1 **LMM+PM** accuracy by taking top-1 **LMM** decoding results as input. It is found that the **PM** decoder is surprisingly effective in that it perfectly reorders 85.9% (51.5/59.9) and 89.1% (36.1 /40.5) target languages in E2C and C2E translation respectively.

All the experiments above recommend that our method is an effective solution for NE translation.

6 Related Work

Since our method has benefited from the JSCM of Li *et al.* [4] and statistical MT research [8-12], let us compare our study with the previous related work.

The *n*-gram JSCM was proposed for machine transliteration by Li *et al.* [4]. It couples the source and channel constraints into a generative model to directly estimate the joint probability of source and target alignment using *n*-gram statistics. It was shown that JSCM captures rich contextual information that is present in a bilingual corpus to model the monotonic generative process of sequential data. In this point, our **LMM** model is the same as JSCM. The only difference is that in machine transliteration Li *et al.* [4] use phoneme unit as the basic modeling unit and our **LMM** is phrase-based.

In our study, we enhance the **LMM** with the **PM** to account for the word reordering issue in NE translation, so our model is capable of modeling the non-monotone problem. In contrast, JSCM only models the monotone problem.

Both rule-based [1] and statistical model-based [5,6] methods have been proposed to address the NE translation problem. The model-based methods mostly are based on conditional probability under the noisy-channel framework [8]. Now let's review the different modeling methods:

1) As far as lexical choice issue is concerned, the noisy-channel model, represented by IBM Model 1-5 [8], models lexical dependency using a context-free conditional probability. Marcu and Wong [10] proposed a phrase-based context-free joint probability model for lexical mapping. In contrast, our **LMM** models lexical dependency using *n*-order bilingual contextual information.
2) Another characteristic of our method lies in its modeling and search strategy. NE translation and MT are usually viewed as a non-monotone search problem and it is well-known that a non-monotone search is exponentially more complex than a monotone search. Thus, we propose the two separated models and the two-step search, so that the lexical mapping issue can be resolved by monotone search. This results in a large improvement on translation selection.
3) In addition, instead of the position-based distortion model [8-12], we use the *n*-gram permutation model to account for word reordering. A risk-free decoder is also proposed for the permutation model.

One may argue that our proposed model bears a strong resemblance to IBM Model 1: a position-independent translation model and a language model on target sentence without explicit distortion modeling. Let us discuss the major differences between them:

1) Our **LMM** models the lexical mapping and target word selection using a context-dependent joint probability while IBM Model 1 using a context-independent conditional probability and a target *n*-gram language model.
2) Our **LMM** carries out the target word selection and our **PM** only models the target word connectivity while the language model in IBM Model 1 performs the function of target word selection.

Alternatively, finite-state automata (FSA) for statistical MT were previous suggested for decoding using contextual information [21,22]. Bangalore and Riccardi [21] proposed a phrase-based variable length *n*-gram model followed by a reordering scheme for spoken language translation. However, their re-ordering scheme was not evaluated by empirical experiments.

7 Conclusions

In this paper, we propose a new model for NE translation. We present the training and decoding methods for the proposed model. We also compare the proposed method with related work. Empirical experiments show that our method outperforms the previous methods significantly in all test cases. We conclude that our method works more effectively and efficiently in NE translation than previous work does.

Our method does well in NE translation, which is relatively less sophisticated in terms of word distortion. We expect to improve its permutation model by integrating a distortion model to account for larger sentence structure and apply to machine translation study.

Acknowledgments

We would like to thank the anonymous reviews for their invaluable suggestions on our original manuscript.

References

1. Hsin-Hsi Chen, Changhua Yang and Ying Lin. 2003. Learning Formulation and Transformation Rules for Multilingual NEs. Proceedings of the ACL 2003 Workshop on MMLNER
2. K. Knight and J. Graehl. 1998. Machine Transliteration. Computational Linguistics, 24(4)
3. Jong-Hoon Oh and Key-Sun Choi, 2002. An English-Korean Transliteration Model Using Pronunciation and Contextual Rules. Proceedings of COLING 2002
4. Haizhou Li, Ming Zhang and Jian Su. 2004. A Joint Source-Channel Model for Machine Transliteration. Proceedings of the 42th ACL, Barcelona, 160-167
5. Y. Al-Onaizan and K. Knight, 2002. Translating named entities using monolingual and bilingual resources. Proceedings of the 40th ACL, Philadelphia, 400-408
6. Fei Huang, S. Vogel and A. Waibel, 2004. Improving NE Translation Combining Phonetic and Semantic Similarities. Proceedings of HLT-NAACL-2004
7. LDC2003E01, 2003. http://www.ldc.upenn.edu/

8. P.F. Brown, S.A.D. Pietra, V.J.D. Pietra and R.L. Mercer.1993. The mathematics of statistical machine translation. Computational Linguistics,19(2):263-313
9. Richard Zens and Hermann Ney. 2004. Improvements in Phrase-Based Statistical Machine Translation. Proceedings of HLT-NAACL-2004
10. D. Marcu and W. Wong. 2002. A Phrase-based, Joint Probability Model for Statistical Machine Translation. Proceedings of EMNLP-2002
11. Franz Joseh Och, C. Tillmann and H. Ney. 1999. Improved Alignment Models for Statistical Machine Translation. Proceedings of Joint Workshop on EMNLP and Very Large Corpus: 20-28
12. P. Koehn, F. J. Och and D. Marcu. 2003. Statistical Phrase-based Translation. Proceedings of HLT-2003
13. A. Stolcke. 2002. SRILM -- An Extensible Language Modeling Toolkit. Proceedings of ICSLP-2002, vol. 2, 901-904, Denver.
14. U. Germann, M. Jahr, K. Knight, D. Marcu and K. Yamada. 2001. Fast Decoding and Optimal Decoding for Machine Translation. Proceedings of ACL-2001
15. Franz Joseh Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. Computational Linguistics, 29(1):19-51
16. U. Germann. 2003. Greedy Decoding for Statistical Machine Translation in Almost Linear Time. Proceedings of HLT-NAACL-2003
17. Christoph Tillmann and Hermann Ney. 2003. Word Reordering and a Dynamic Programming Beam Search Algorithm for Statistical Machine Translation. Computational Linguistics, 29(1):97-133
18. R. Schwartz and Y. L. Chow. 1990. The N-best algorithm: An efficient and Exact procedure for finding the N most likely sentence hypothesis, Proceedings of ICASSP 1990, 81-84
19. K. Papineni, S. Roukos, T. Ward and W. J. Zhu. 2001. BLEU: a method for automatic evaluation of machine translation. Technical Report RC22176 (W0109-022), IBM Research Report.
20. G. Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. Proceedings of ARPA Workshop on HLT
21. S. Bangalore and G. Riccardi, 2000, Stochastic Finite State Models for Spoken Language Machine Translation, Workshop on Embedded MT System
22. Stephan Kanthak and Hermann Hey, 2004. FSA: An Efficient and Flexiable C++ Tookkit for Finite State Automata Using On-Demand Computation, Proceedings of ACL-2004