Nonparametric Bayesian Machine Transliteration with Synchronous Adaptor Grammars

Yun Huang[1,2] huangyun@comp.nus.edu.sg
Min Zhang[1] mzhang@i2r.a-star.edu.sg
Chew Lim Tan[2] tancl@comp.nus.edu.sg

[1]Human Language Department
Institute for Infocomm Research
1 Fusionopolis Way, Singapore

[2]Department of Computer Science
National University of Singapore
13 Computing Drive, Singapore

Abstract

Machine transliteration is defined as automatic phonetic translation of names across languages. In this paper, we propose synchronous adaptor grammar, a novel nonparametric Bayesian learning approach, for machine transliteration. This model provides a general framework without heuristic or restriction to automatically learn syllable equivalents between languages. The proposed model outperforms the state-of-the-art EM-based model in the English to Chinese transliteration task.

1 Introduction

Proper names are one source of OOV words in many NLP tasks, such as machine translation and cross-lingual information retrieval. They are often translated through transliteration, i.e. translation by preserving how words sound in both languages. In general, machine transliteration is often modelled as monotonic machine translation (Rama and Gali, 2009; Finch and Sumita, 2009; Finch and Sumita, 2010), the joint source-channel models (Li et al., 2004; Yang et al., 2009), or the sequential labeling problems (Reddy and Waxmonsky, 2009; Abdul Hamid and Darwish, 2010).

Syllable equivalents acquisition is a critical phase for all these models. Traditional learning approaches aim to maximize the likelihood of training data by the Expectation-Maximization (EM) algorithm. However, the EM algorithm may over-fit the training data by memorizing the whole training instances. To avoid this problem, some approaches restrict that a single character in one language could be aligned to many characters of the other, but not vice versa (Li et al., 2004; Yang et al., 2009). Heuristics are introduced to obtain many-to-many alignments by combining two directional one-to-many alignments (Rama and Gali, 2009). Compared to maximum likelihood approaches, Bayesian models provide a systemic way to encode knowledges and infer compact structures. They have been successfully applied to many machine learning tasks (Liu and Gildea, 2009; Zhang et al., 2008; Blunsom et al., 2009).

Among these models, Adaptor Grammars (AGs) provide a framework for defining nonparametric Bayesian models based on PCFGs (Johnson et al., 2007). They introduce additional stochastic processes (named *adaptors*) allowing the expansion of an adapted symbol to depend on the expansion history. Since many existing models could be viewed as special kinds of PCFG, adaptor grammars give general Bayesian extension to them. AGs have been used in various NLP tasks such as topic modeling (Johnson, 2010), perspective modeling (Hardisty et al., 2010), morphology analysis and word segmentation (Johnson and Goldwater, 2009; Johnson, 2008).

In this paper, we extend AGs to Synchronous Adaptor Grammars (SAGs), and describe the inference algorithm based on the Pitman-Yor process (Pitman and Yor, 1997). We also describe how transliteration could be modelled under this formalism. It should be emphasized that the proposed method is language independent and heuristic-free. Experiments show the proposed approach outperforms the strong EM-based baseline in the English to Chinese transliteration task.

2 Synchronous Adaptor Grammars

2.1 Model

A Pitman-Yor Synchronous Adaptor Grammar (PYSAG) is a tuple $\mathcal{G} = (\mathcal{G}_s, \mathcal{N}_a, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{\alpha})$, where $\mathcal{G}_s = (\mathcal{N}, \mathcal{T}_s, \mathcal{T}_t, \mathcal{R}, S, \boldsymbol{\Theta})$ is a Synchronous Context-Free Grammar (SCFG) (Chiang, 2007), $\mathcal{N}$ is a set of nonterminal symbols, $\mathcal{T}_s/\mathcal{T}_t$ are source/target terminal symbols, $\mathcal{R}$ is a set of rewrite rules, $S \in \mathcal{N}$ is the start symbol, $\boldsymbol{\Theta}$ is the distribution of rule probabilities, $\mathcal{N}_a \subseteq \mathcal{N}$ is the set of adapted nonterminals, $\boldsymbol{a} \in [0,1], \boldsymbol{b} \geq 0$ are vectors of discount and concentration parameters both indexed by adapted nonterminals, and $\boldsymbol{\alpha}$ are Dirichlet prior parameters.

Algorithm 1 Generative Process

```
1:  draw $\boldsymbol{\theta}_A \sim \mathrm{Dir}(\alpha_A)$ for all $A \in \mathcal{N}$
2:  for each yield pair $\langle s \;/\; t \rangle$ do
3:      SAMPLE($S$)                              ▷ Sample from root
4:  return

5:  function SAMPLE($A$)                         ▷ For $A \in \mathcal{N}$
6:      if $A \in \mathcal{N}_a$ then
7:          return SAMPLESAG($A$)
8:      else
9:          return SAMPLESCFG($A$)

10: function SAMPLESCFG($A$)                     ▷ For $A \notin \mathcal{N}_a$
11:     draw rule $r = \langle \beta \;/\; \gamma \rangle \sim \mathrm{Multi}(\boldsymbol{\theta}_A)$
12:     tree $t_B \leftarrow$SAMPLE($B$) for nonterminal $B \in \beta \cup \gamma$
13:     return BUILDTREE($r, t_{B_1}, t_{B_2}, \ldots$)

14: function SAMPLESAG($A$)                      ▷ For $A \in \mathcal{N}_a$
15:     draw cache index $z_{n+1} \sim P(z|z_{i<n})$, where
        $P(z|z_{i<n}) = \begin{cases} \frac{ma+b}{n+b} & \text{if } z_{n+1} = m+1 \\ \frac{n_k-a}{n+b} & \text{if } z_{n+1} = k \in \{1, \cdots, m\} \end{cases}$
16:     if $z_{n+1} = m+1$ then                  ▷ New entry
17:         tree $t \leftarrow$ SAMPLESCFG($A$)
18:         $m \leftarrow m+1$; $n_m = 1$          ▷ Update counts
19:         INSERTTOCACHE($\mathcal{C}_A, t$).
20:     else                                     ▷ Old entry
21:         $n_k \leftarrow n_k + 1$
22:         tree $t \leftarrow$ FINDINCACHE($\mathcal{C}_A, z_{n+1}$)
23:     return $t$
```

The generative process of a synchronous tree set $\boldsymbol{T}$ is described in Algorithm 1. First, rule probabilities are sampled for each nonterminal $A \in \mathcal{N}$ (line 1) according to the Dirichlet distribution. Then synchronous trees are generated in the top-down fashion from the start symbol S (line 3) for each yield pair. For nonterminals that are not adapted, the grammar expands it just as the original synchronous grammar (function SAMPLESCFG). For each adapted nonterminal $A \in \mathcal{N}_a$, the grammar maintains a cache $\mathcal{C}_A$ to store previously generated subtrees under A. Let z_i be the subtree index in $\mathcal{C}_A$, denoting the synchronous subtree generated at the i^{th} expansion of A. At some particular time, assuming n subtrees rooted at A have been generated with m different types in the cache of A, each of which has been generated for $n_1, \ldots, n_m$ times respectively[1]. Then the grammar either generates the $(n+1)^{th}$ synchronous subtree as SCFG (line 17) or chooses an existing subtree (line 22), according to the conditional probability $P(z|z_{i<n})$.

The above generative process demonstrates "rich get richer" dynamics, i.e. previous sampled subtrees under adapted nonterminals would more likely be sampled again in following procedures. This is suitable for many learning tasks since they prefer sparse solutions to avoid the over-fitting problems. If we integrate out the adaptors, the joint probability of a particular sequence of indexes $\boldsymbol{z}$ with cached counts $(n_1, \ldots, n_m)$ under the Pitman-Yor process is

$$PY(\boldsymbol{z}|a,b) = \frac{\prod_{k=1}^{m}(a(k-1)+b)\prod_{j=1}^{n_k-1}(j-a)}{\prod_{i=0}^{n-1}(i+b)}. \quad (1)$$

Given synchronous tree set $\boldsymbol{T}$, the joint probability under the PYSAG is

$$P(\boldsymbol{T}|\boldsymbol{\alpha},\boldsymbol{a},\boldsymbol{b}) = \prod_{A \in \mathcal{N}} \frac{\mathrm{B}(\boldsymbol{\alpha}_A + \boldsymbol{f}_A)}{\mathrm{B}(\boldsymbol{\alpha}_A)} PY(\boldsymbol{z}(\boldsymbol{T})|\boldsymbol{a},\boldsymbol{b}) \quad (2)$$

where $\boldsymbol{f}_A$ is the vector containing the number of times that rules $r \in \mathcal{R}_A$ are used in the $\boldsymbol{T}$, and B is the Beta function.

2.2 Inference for PYSAGs

Directly drawing samples from Equation (2) is intractable, so we extend the component-wise Metropolis-Hastings algorithm (Johnson et al., 2007) to the synchronous case. In detail, we draw sample T_i' from some proposal distribution $Q(T_i|y_i, \boldsymbol{T}_{-i})$[2], then accept the new sampled syn-

[1]Obviously, $n = \sum_{k=1}^{m} n_k$.

[2]$\boldsymbol{T}_{-i}$ means the set of sampled trees except the i^{th} one.

chronous tree T_i' with probability

$$A(T_i, T_i') = \min\left\{1, \frac{P(\boldsymbol{T}'|\boldsymbol{\alpha}, \boldsymbol{a}, \boldsymbol{b})Q(T_i|y_i, \boldsymbol{T}_{-i})}{P(\boldsymbol{T}|\boldsymbol{\alpha}, \boldsymbol{a}, \boldsymbol{b})Q(T_i'|y_i, \boldsymbol{T}_{-i})}\right\}. \tag{3}$$

In theory, Q could be any distribution if it never assigns zero probability. For efficiency reason, we choose the probabilistic SCFG as the proposal distribution. We pre-parse the training instances[3] before inference and save the structure of synchronous parsing forests. During the inference, we only change rule probabilities in parsing forests without changing the forest structures. The probability of rule $r \in \mathcal{R}_A$ in Q is estimated by relative frequency $\theta_r = \frac{[f_r]_{-i}}{\sum_{r' \in \mathcal{R}_A} [f_{r'}]_{-i}}$, where $\mathcal{R}_A$ is the set of rules rooted at A, and $[f_r]_{-i}$ is the number of times that rule r is used in the tree set $\boldsymbol{T}_{-i}$. We use the sampling algorithm described in (Blunsom and Osborne, 2008) to draw a synchronous tree from the parsing forest according to the proposal Q.

Following (Johnson and Goldwater, 2009), we put an uninformative Beta(1,1) prior on $\boldsymbol{a}$ and a "vague" Gamma(10, 0.1) prior on $\boldsymbol{b}$ to model the uncertainty of hyperparameters.

3 Machine Transliteration

3.1 Grammars

For machine transliteration, we design the following grammar to learn syllable mappings[4]:

$$\begin{array}{rcl}
\texttt{Name} & \rightarrow & \langle \underline{\texttt{Syl}} \ / \ \underline{\texttt{Syl}} \rangle^+ \\
\underline{\texttt{Syl}} & \rightarrow & \langle \texttt{NECs} \ / \ \texttt{NECs} \rangle \\
\underline{\texttt{Syl}} & \rightarrow & \langle \texttt{NECs SECs} \ / \ \texttt{NECs SECs} \rangle \\
\underline{\texttt{Syl}} & \rightarrow & \langle \texttt{NECs TECs} \ / \ \texttt{NECs TECs} \rangle \\
\texttt{NECs} & \rightarrow & \langle \texttt{NEC} \ / \ \texttt{NEC} \rangle^+ \\
\texttt{SECs} & \rightarrow & \langle \texttt{SEC} \ / \ \texttt{SEC} \rangle^+ \\
\texttt{TECs} & \rightarrow & \langle \texttt{TEC} \ / \ \texttt{TEC} \rangle^+ \\
\texttt{NEC} & \rightarrow & \langle s_i \ / \ t_j \rangle \\
\texttt{SEC} & \rightarrow & \langle \varepsilon \ / \ t_j \rangle \\
\texttt{TEC} & \rightarrow & \langle s_i \ / \ \varepsilon \rangle
\end{array}$$

[3]We implement the CKY-like bottom up parsing algorithm described in (Wu, 1997). The complexity is $O(|s|^3|t|^3)$.

[4]Similar to (Johnson, 2008), the adapted nonterminal are underlined. Similarly, we also use rules in the regular expression style $\texttt{X} \rightarrow \langle \texttt{A} \ / \ \texttt{A} \rangle^+$ to denote the following three rules:

$$\begin{array}{rcl}
\texttt{X} & \rightarrow & \langle \texttt{As} \ / \ \texttt{As} \rangle \\
\texttt{As} & \rightarrow & \langle \texttt{A} \ / \ \texttt{A} \rangle \\
\texttt{As} & \rightarrow & \langle \texttt{A As} \ / \ \texttt{A As} \rangle
\end{array}$$

where the adapted nonterminal $\underline{\texttt{Syl}}$ is designed to capture the syllable equivalents between two languages, and the nonterminal NEC, SEC and TEC capture the character pairs with no empty character, empty source and empty target respectively. Note that this grammar restricts the leftmost characters on both sides must be aligned one-by-one. Since our goal is to learn the syllable equivalents, we are not interested in the subtree tree inside the syllables. We refer this grammar as *syllable grammar*.

The above grammar could capture inner-syllable dependencies. However, the selection of the target characters also depend on the context. For example, the following three instances are found in the training set:

⟨a a b y e / 奥[ao] 比[bi]⟩
⟨a a g a a r d / 埃[ai] 格[ge] 德[de]⟩
⟨a a l t o / 阿[a] 尔[er] 托[tuo]⟩

where the same English syllable ⟨a a⟩ are transliterated to ⟨奥[ao]⟩, ⟨埃[ai]⟩ and ⟨阿[a]⟩ respectively, depending on the following syllables. To model these contextual dependencies, we propose the hierarchical SAG. The two-layer *word grammar* is obtained by adding following rules:

$$\begin{array}{rcl}
\texttt{Name} & \rightarrow & \langle \underline{\texttt{Word}} \ / \ \underline{\texttt{Word}} \rangle^+ \\
\underline{\texttt{Word}} & \rightarrow & \langle \underline{\texttt{Syl}} \ / \ \underline{\texttt{Syl}} \rangle^+
\end{array}$$

We might further add a new adapted nonterminal $\underline{\texttt{Col}}$ to learn the word collocations. The following rules appear in the *collocation grammar*:

$$\begin{array}{rcl}
\texttt{Name} & \rightarrow & \langle \underline{\texttt{Col}} \ / \ \underline{\texttt{Col}} \rangle^+ \\
\underline{\texttt{Col}} & \rightarrow & \langle \underline{\texttt{Word}} \ / \ \underline{\texttt{Word}} \rangle^+ \\
\underline{\texttt{Word}} & \rightarrow & \langle \underline{\texttt{Syl}} \ / \ \underline{\texttt{Syl}} \rangle^+
\end{array}$$

Figure 1 gives one synchronous parsing trees under the collocation grammar of the example ⟨m a x / 麦[mai] 克[ke] 斯[si]⟩.

3.2 Translation Model

After sampling, we need a translation model to transliterate new source string to target string. Following (Li et al., 2004), we use the n-gram translation model to estimate the joint distribution $P(s,t) = \prod_{k=1}^{K} P(p_k|p_1^{k-1})$, where p_k is the k^{th} syllable pair of the string pair $\langle s \ / \ t \rangle$.

The first step is to construct joint segmentation lattice for each training instance. We first generate a merged grammar G' using collected subtrees under adapted nonterminals, then use synchronous parsing

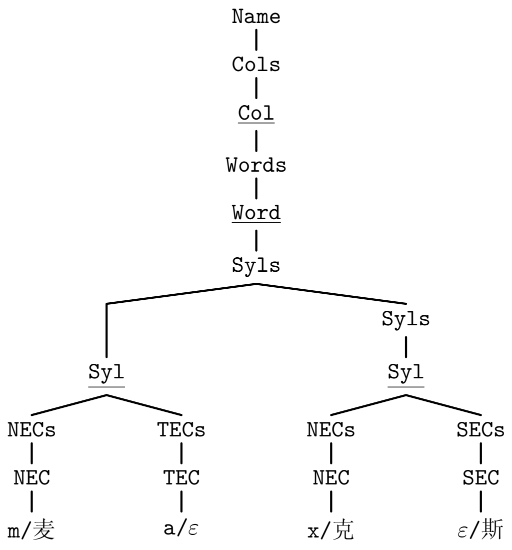

Figure 1: An example of parse tree.

to obtain probabilities in the segmentation lattice. Specifically, we "flatten" the collected subtrees under Syl, i.e. removing internal nodes, to construct new synchronous rules. For example, we could get two rules from the tree in Figure 1:

$$\begin{aligned}\underline{\texttt{Syl}} &\rightarrow \langle \texttt{m a} \,/\, \text{麦}\rangle \\ \underline{\texttt{Syl}} &\rightarrow \langle \texttt{x} \,/\, \text{克斯}\rangle\end{aligned}$$

If multiple subtrees are flattened to the same synchronous rule, we sum up the counts of these subtrees. For rules with non-adapted nonterminal as parent, we assign the probability as the same of the sampled rule probability, i.e. let $\theta'_r = \theta_r$. For the adapted nonterminal Syl, there are two kinds of rules: (1) the rules in the original probabilistic SCFG, and (2) the rules flattened from subtrees. We assign the rule probability as

$$\theta'_r = \begin{cases} \frac{ma+b}{n+b} \cdot \theta_r & \text{if } r \text{ is original SCFG rule} \\ \frac{n_r - a}{n+b} & \text{if } r \text{ is flatten from subtree} \end{cases} \quad (4)$$

where a and b are the parameters associated with Syl, m is the number of types of different rules flatten from subtrees, n_r is the count of rule r, and n is the total number of flatten rules. One may verify that the rule probabilities are well normalized. Based on this merged grammar G', we parse the training string pairs, then encode the parsed forest into the lattice. Figure 2 show a lattice example for the string pair ⟨a a l t o / 阿[a] 尔[er] 托[tuo]⟩. The transition probabilities in the lattice are the "inside" probabilities of corresponding Syl node in the parsing forest.

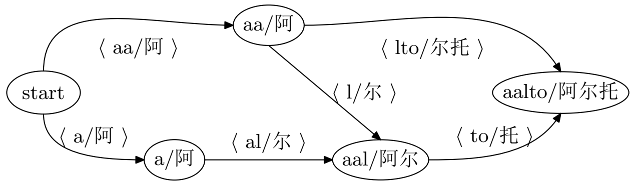

Figure 2: Lattice example.

After building the segmentation lattice, we train 3-order language model from the lattice using the SRILM[5]. In decoding, given a new source string, we use the Viterbi algorithm with beam search (Li et al., 2004) to find the best transliteration candidate.

4 Experiments

4.1 Data and Settings

We conduct experiments on the English-Chinese data in the ACL Named Entities Workshop (NEWS 2009) [6]. Table 1 gives some statistics of the data. For evaluation, we report the *word accuracy* and *mean F-score* metrics defined in (Li et al., 2009).

	Train	Dev	Test
# Entry	31,961	2,896	2,896
# En Char	218,073	19,755	19,864
# Ch Char	101,205	9,160	9,246
# Ch Type	370	275	283

Table 1: Transliteration data statistics

In the inference step, we first run sampler through the whole training corpus for 10 iterations, then collect adapted subtree statistics for every 10 iterations, and finally stop after 20 collections. After each iteration, we resample each of hyperparameters from the posterior distribution of hyperparameters using a slice sampler (Neal, 2003).

4.2 Results

We implement the joint source-channel model (Li et al., 2004) as the baseline system, in which the orthographic syllable alignment is automatically derived by the Expectation-Maximization (EM) algorithm.

[5]http://www.speech.sri.com/projects/srilm/

[6]http://www.acl-ijcnlp-2009.org/workshops/NEWS2009/

Since EM tends to memorize the training instance as a whole, Li et al. (2004) restrict the Chinese side to be single character in syllable equivalents. Our method can be viewed as the Bayesian extension of the EM-based baseline. Since PYSAGs could learn accurate and compact transliteration units, we do not need the restriction any more.

Grammar	Dev (%)	Test (%)
Baseline	67.8/86.9	66.6/85.7
Syl	66.6/87.0	66.6/86.6
Word	67.1/87.2	**67.0/86.7**
Col	67.2/87.1	66.9/86.7

Table 2: Transliteration results, in the format of *word accuracy / mean F-score*. "Syl","Word" and "Col" denote the syllable, word and collocation grammar respectively.

Table 2 presents the results of all experiments. From this table, we draw following conclusions:

1. The best results of our model are $67.1\%/87.2\%$ on development set and corresponding $67.0\%/86.7\%$ on test set, achieved by word grammars. The results on test set outperform the EM-based baseline system on both word accuracy and mean F-score.
2. Comparing grammars of different layers, we find that the word grammars perform consistently better than the syllable grammars. These support the assumption that the context information are helpful to identify syllable equivalents. However, the collocation grammars do not further improve performance. We guess the reason is that the instances in transliteration are very short, so two-layer grammars are good enough while the collocations become very sparse, which results in unreliable probability estimation.

4.3 Discussion

Table 3 shows some examples of learned syllable mappings in the final sampled tree of the syllable grammar. We can see that the PYSAGs could find good syllable mappings from the raw name pairs without any heuristic or restriction. In this point of view, the proposed method is language independent.

Specifically, we are interested in the English token "x", which is the only one that has two corresponding Chinese characters ("克 斯[ke si]"). Table 3 demonstrates that nearly all these correct mappings are discovered by PYSAGs. Note that these kinds of mapping can not be learned if we restrict the Chinese side to be only one character (the heuristic used in (Li et al., 2004)). We will conduct experiments on other language pairs in the future.

s/斯[si]/1669	k/克[ke]/408	ri/里[li]/342
t/特[te]/728	ma/马[ma]/390	ra/拉[la]/339
man/曼[man]/703	co/科[ke]/387	ca/卡[ka]/333
d/德[de]/579	ll/尔[er]/383	m/姆[mu]/323
ck/克[ke]/564	la/拉[la]/382	li/利[li]/314
de/德[de]/564	tt/特[te]/380	ber/伯[bo]/311
ro/罗[luo]/531	l/尔[er]/367	ley/利[li]/310
son/森[sen]/442	ton/顿[dun]/360	na/纳[na]/302
x/克斯[ke si]/40	x/克[ke]/3	x/斯[si]/1

Table 3: Examples of learned syllable mappings. Chinese Pinyin are given in the square bracket. The counts of syllable mappings in the final sampled tree are also given.

5 Conclusion

This paper proposes synchronous adaptor grammars, a nonparametric Bayesian model, for machine transliteration. Based on the sampling, the PYSAGs could automatically discover syllable equivalents without any heuristic or restriction. In this point of view, the proposed model is language independent. The joint source-channel model is then used for training and decoding. Experimental results on the English-Chinese transliteration task show that the proposed method outperforms the strong EM-based baseline system. We also compare grammars in different layers and find that the two-layer grammars are suitable for the transliteration task. We plan to carry out more transliteration experiments on other language pairs in the future.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments and suggestions. We also thank Zhixiang Ren, Zhenghua Li, and Jun Sun for insightful discussions. Special thanks to Professor Mark Johnson for his open-source codes[7].

[7]Available from `http://web.science.mq.edu.au/~mjohnson/Software.htm`

References

Ahmed Abdul Hamid and Kareem Darwish. 2010. Simplified feature set for arabic named entity recognition. In *Proceedings of the 2010 Named Entities Workshop*, pages 110–115, Uppsala, Sweden, July.

Phil Blunsom and Miles Osborne. 2008. Probabilistic inference for machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 215–223, Honolulu, Hawaii, October.

Phil Blunsom, Trevor Cohn, Chris Dyer, and Miles Osborne. 2009. A gibbs sampler for phrasal synchronous grammar induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 782–790, Suntec, Singapore, August.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228, June.

Andrew Finch and Eiichiro Sumita. 2009. Transliteration by bidirectional statistical machine translation. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 52–56, Suntec, Singapore, August.

Andrew Finch and Eiichiro Sumita. 2010. A Bayesian Model of Bilingual Segmentation for Transliteration. In *Proceedings of the 7th International Workshop on Spoken Language Translation (IWSLT)*, pages 259–266, Paris, France, December.

Eric Hardisty, Jordan Boyd-Graber, and Philip Resnik. 2010. Modeling perspective using adaptor grammars. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 284–292, Cambridge, MA, October.

Mark Johnson and Sharon Goldwater. 2009. Improving nonparameteric bayesian inference: experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 317–325, Boulder, Colorado, June.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007. Adaptor grammars: A framework for specifying compositional nonparametric bayesian models. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 641–648. Cambridge, MA.

Mark Johnson. 2008. Using adaptor grammars to identify synergies in the unsupervised acquisition of linguistic structure. In *Proceedings of ACL-08: HLT*, pages 398–406, Columbus, Ohio, June.

Mark Johnson. 2010. Pcfgs, topic models, adaptor grammars and learning topical collocations and the structure of proper names. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1148–1157, Uppsala, Sweden, July.

Haizhou Li, Min Zhang, and Jian Su. 2004. A joint source-channel model for machine transliteration. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 159–166, Barcelona, Spain, July.

Haizhou Li, A Kumaran, Vladimir Pervouchine, and Min Zhang. 2009. Report of news 2009 machine transliteration shared task. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 1–18, Suntec, Singapore, August.

Ding Liu and Daniel Gildea. 2009. Bayesian learning of phrasal tree-to-string templates. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1308–1317, Singapore, August.

Radford M. Neal. 2003. Slice sampling. *Annals of Statistics*, 31(3):705–767.

J. Pitman and M. Yor. 1997. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Annals of Probability*, 25:855–900.

Taraka Rama and Karthik Gali. 2009. Modeling machine transliteration as a phrase based statistical machine translation problem. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 124–127, Suntec, Singapore, August.

Sravana Reddy and Sonjia Waxmonsky. 2009. Substring-based transliteration with conditional random fields. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 92–95, Suntec, Singapore, August.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403, September.

Dong Yang, Paul Dixon, Yi-Cheng Pan, Tasuku Oonishi, Masanobu Nakamura, and Sadaoki Furui. 2009. Combining a two-step conditional random field model and a joint source channel model for machine transliteration. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 72–75, Suntec, Singapore, August.

Hao Zhang, Chris Quirk, Robert C. Moore, and Daniel Gildea. 2008. Bayesian learning of non-compositional phrases with synchronous parsing. In *Proceedings of ACL-08: HLT*, pages 97–105, Columbus, Ohio, June.