Log-linear Models for Word Alignment

Yang Liu, Qun Liu and Shouxun Lin

Institute of Computing Technology
Chinese Academy of Sciences

Categories of Alignment Approaches

- Statistical approaches
  - based on well-founded probabilistic models
  - depend on unknown parameters that are learned from training data
- Heuristic approaches
  - use various similarity functions between the types of two languages

Previous Work

- Combination of association clues (Tiedemann, 2003)
- Model 6, a log-linear combination of IBM Model 4 and HMM model (Och and Ney, 2003)
- A probability model, allowing easy integration of context-specific features (Cherry and Lin, 2003)

Log-linear models

$$\Pr(x \mid y) = \frac{\exp\left\{\sum_{m=1}^{M} l_m h_m(x, y)\right\}}{\sum_{x'} \exp\left\{\sum_{m=1}^{M} l_m h_m(x', y)\right\}}$$

Log-linear models, which are very suitable to incorporate additional dependencies, have been successfully applied to statistical machine translation (Och and Ney, 2002).

Log-linear Models for Word Alignment

$$\Pr(\mathrm{a}|\mathrm{e},\mathrm{f}) = \frac{\exp\left\{\sum_{m=1}^{M} l_m h_m(\mathrm{a},\mathrm{e},\mathrm{f})\right\}}{\sum_{\mathrm{a}'} \exp\left\{\sum_{m=1}^{M} l_m h_m(\mathrm{a}',\mathrm{e},\mathrm{f})\right\}}$$

Log-linear models ARE statistical models.

Three Problems

- Feature selection
  - Which knowledge sources are useful and how to design feature functions to make use of them?
- Training
  - How to estimate the model scaling factors?
- Search
  - How to search the optimal alignment in an effective and efficient way?

Feature selection

- IBM translation model 3

$$h(\mathbf{a},\mathbf{e},\mathbf{f}) = Pr(f_1^J, a_1^J | e_1^I)$$
$$= \binom{m-\phi_0}{\phi_0} p_0{}^{m-2\phi_0} p_1{}^{\phi_0} \prod_{i=1}^{l} \phi_i! n(\phi_i|e_i) \times$$
$$\prod_{j=1}^{m} t(f_j|e_{a_j}) d(j|a_j, l, m)$$

- POS tags transition model

$$h(\mathbf{a},\mathbf{e},\mathbf{f},\mathbf{eT},\mathbf{fT}) = \prod_a t(fT_{a(j)}|eT_{a(i)})$$

- Bilingual dictionary coverage

$$h(\mathbf{a},\mathbf{e},\mathbf{f},\mathbf{D}) = \sum_a occur(e_{a(i)}, f_{a(j)}, D)$$

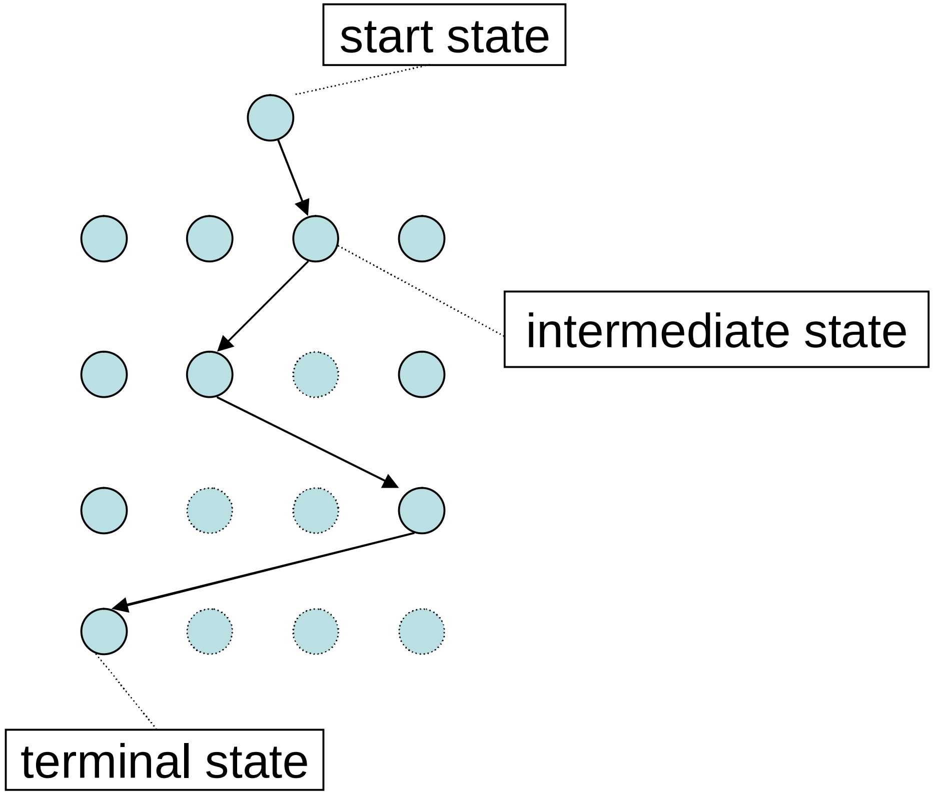
Training

- We use YASMET, which implement GIS algorithm, to train model scaling factors.
- We select the model parameters that yield best alignments on the development corpus
- POS tags transition probabilities are also estimated on development corpus

$$p(fT|eT) = \frac{N_A(fT, eT)}{N(eT)}$$

Here, $N_A(fT, eT)$ is the frequency that the POS tag fT is aligned to POS tag eT and $N(eT)$ is the frequency of eT in the development corpus.

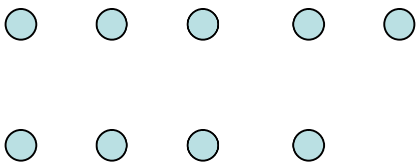
Search



We use a greedy search algorithm to search the alignment with highest probability in the space of all possible alignments. A state in this space is a partial alignment. A transition is defined as the addition of a single link to the current state. A start state is the empty alignment. A terminal state is a state in which no more links can be added to increase the probability of current state.

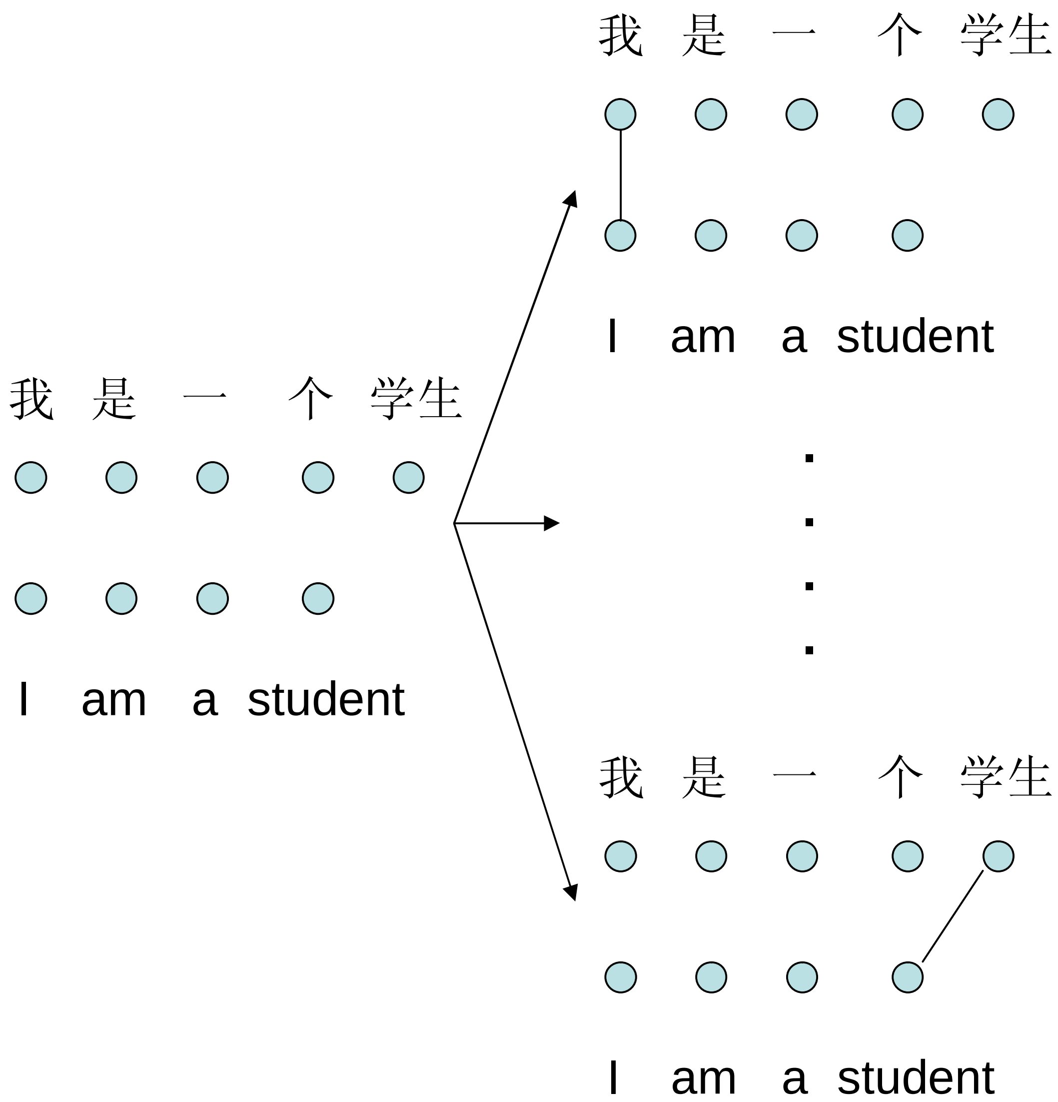
An Example

我　是　一　个　学生



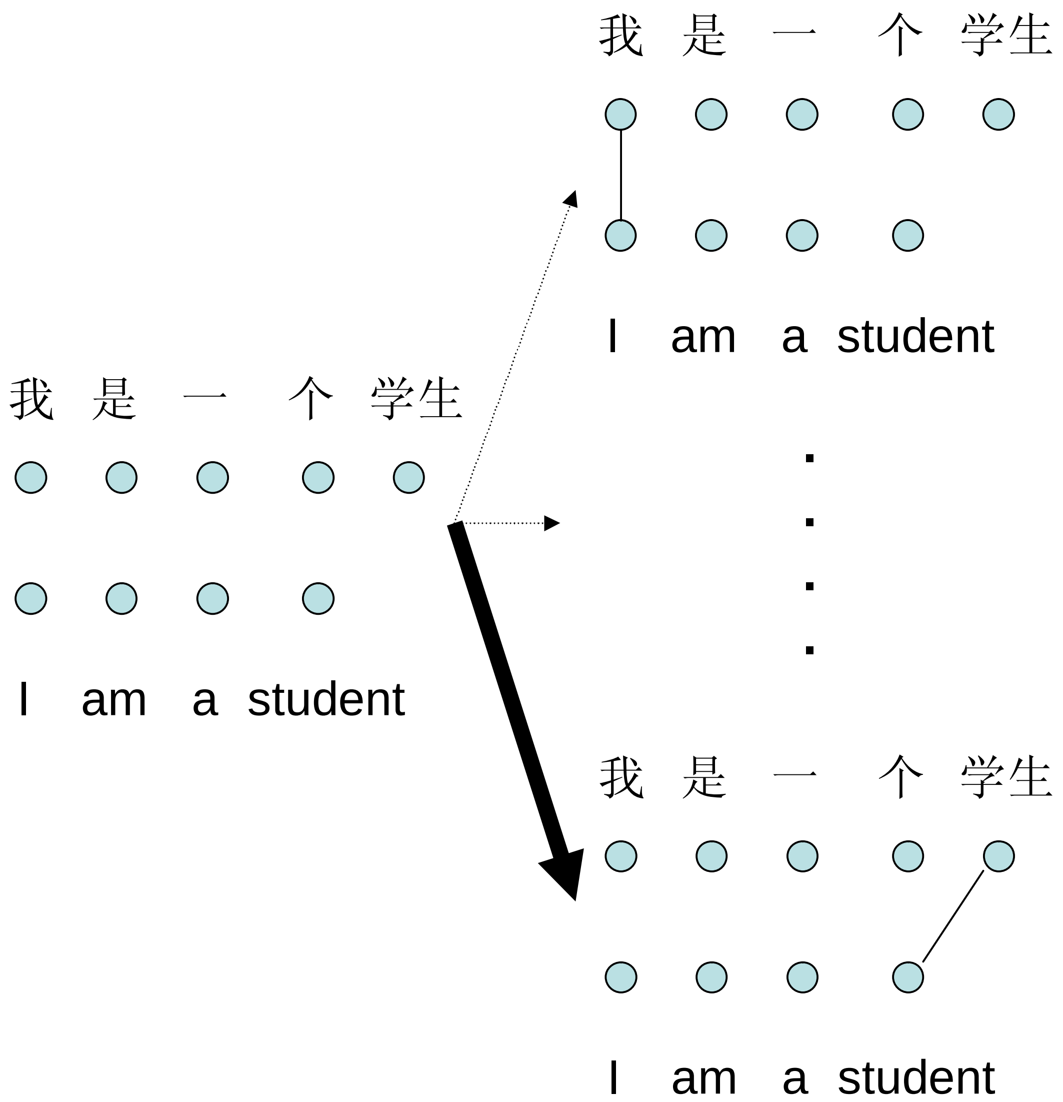
I　am　a　student

An Example



20 possible links!

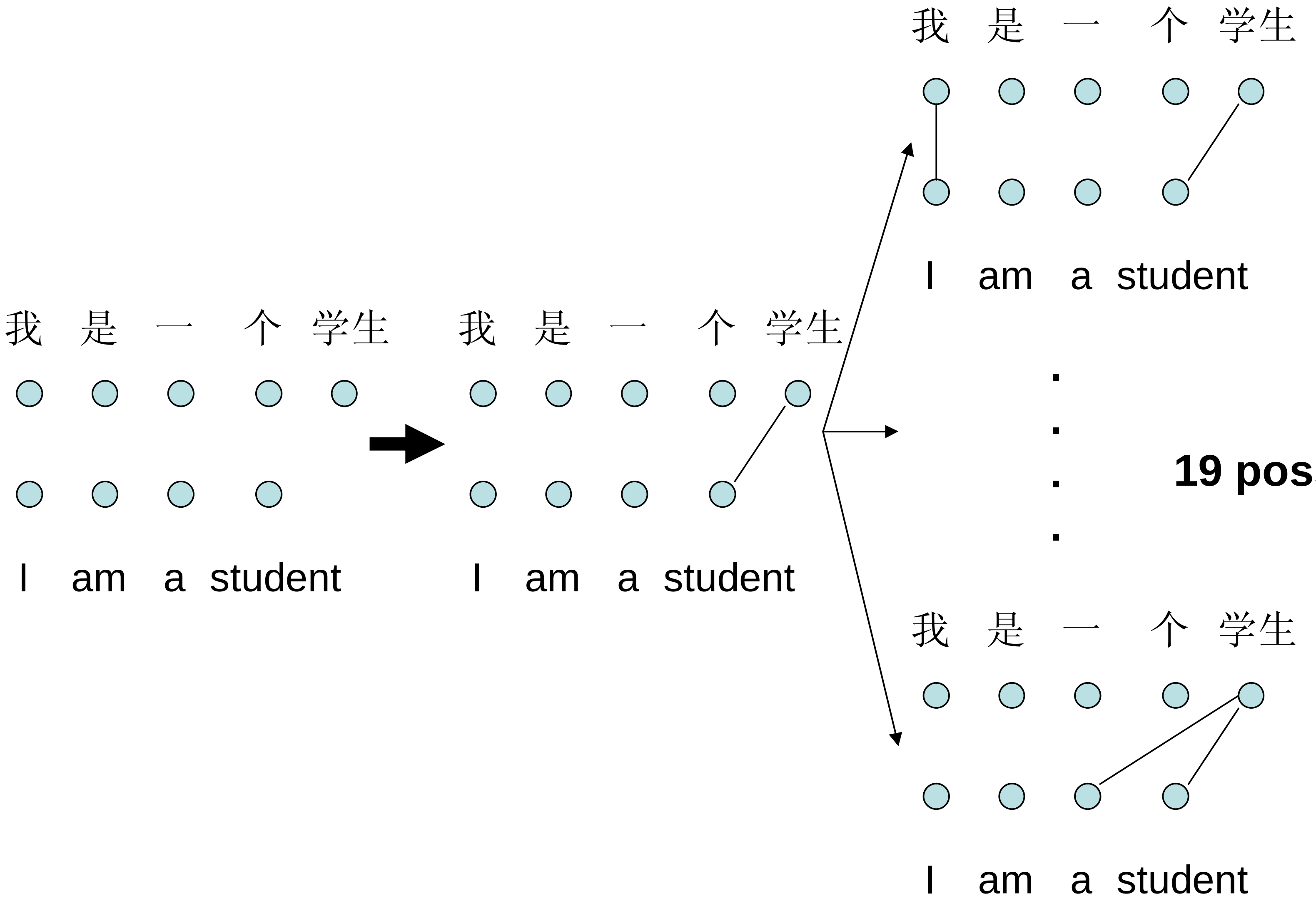
An Example



20 possible links!

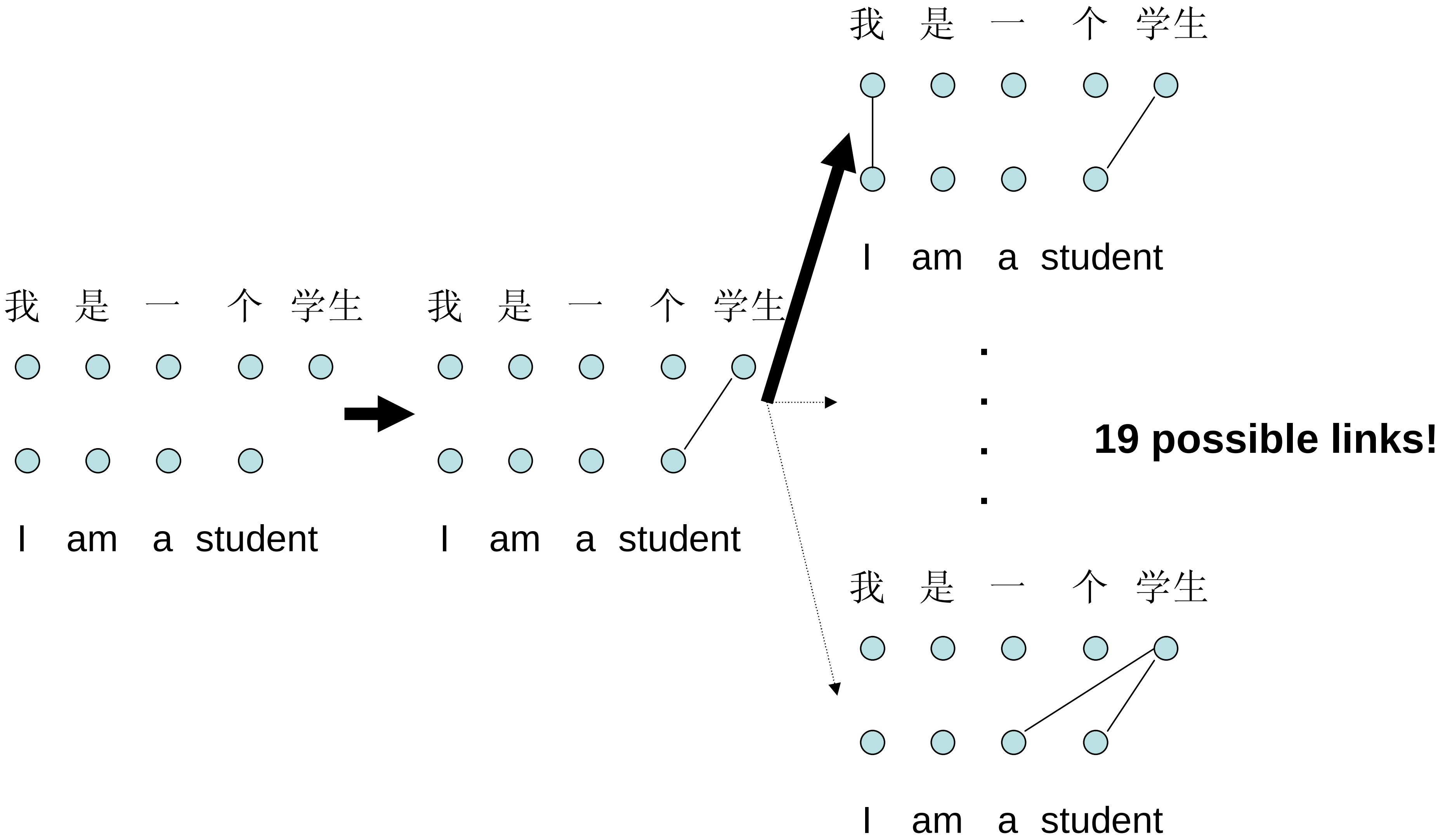
The partial alignment with the greatest probability

An Example



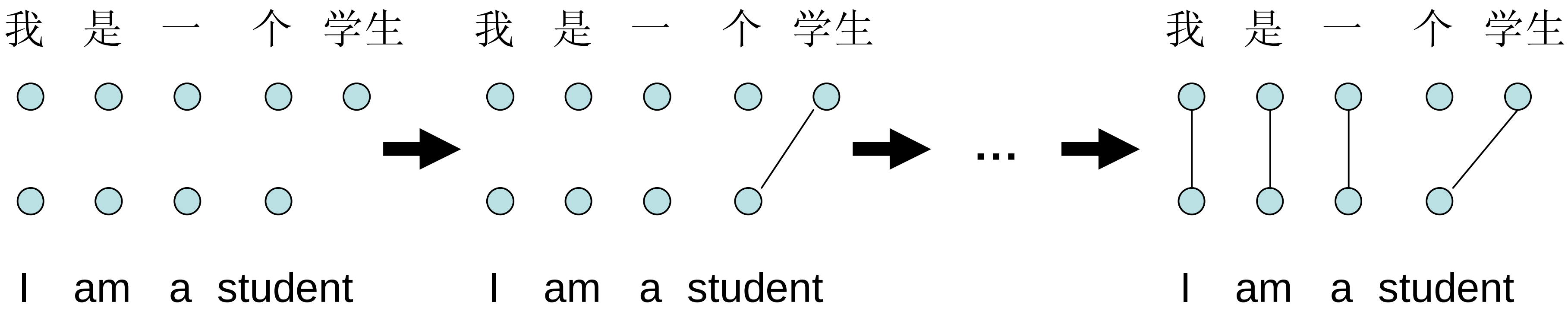
19 possible links!

An Example



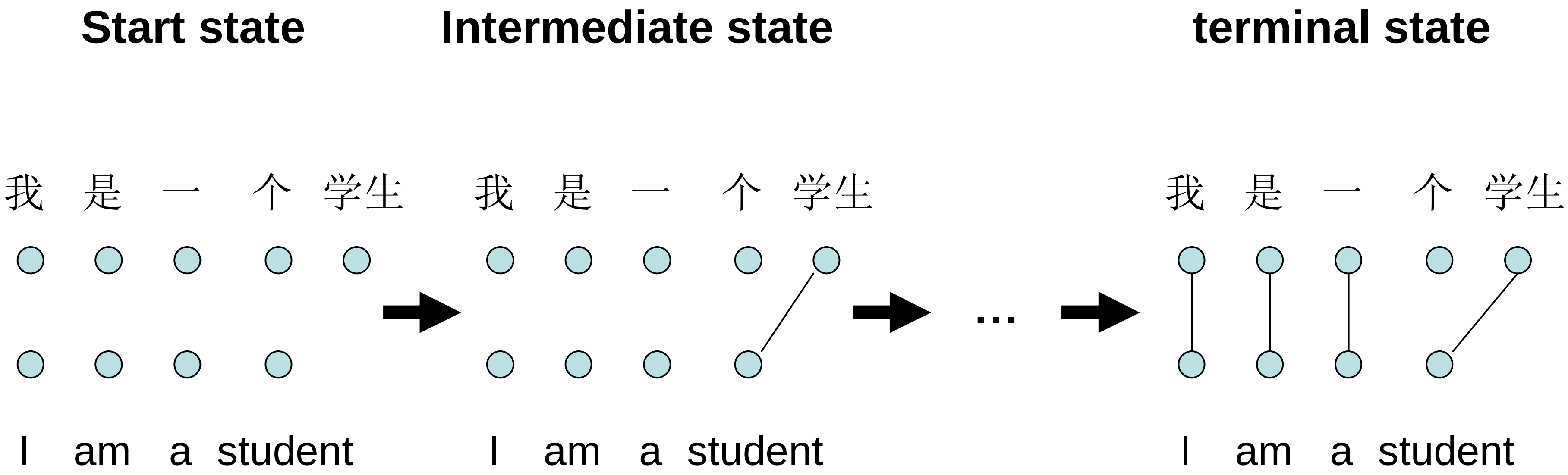
我 是 一 个 学生
I am a student
我 是 一 个 学生
I am a student
我 是 一 个 学生
I am a student
我 是 一 个 学生
I am a student
19 possible links!

An Example



我 是 一 个 学生
I am a student
我 是 一 个 学生
I am a student
...
我 是 一 个 学生
I am a student

An Example



No links can be added to increase the probability of terminal state!

Gain

$$gain(\mathbf{a}, l) = \frac{\exp\left\{\sum_{m=1}^{M} \lambda_m h_m(\mathbf{a} \cup l, \mathbf{e}, \mathbf{f})\right\}}{\exp\left\{\sum_{m=1}^{M} \lambda_m h_m(\mathbf{a}, \mathbf{e}, \mathbf{f})\right\}}$$

We compute *gain*, which is a heuristic function, instead of probability for efficiency.

Search Algorithm

Input: $\mathbf{e}, \mathbf{f}, \mathbf{eT}, \mathbf{fT}$, and $\mathbf{D}$

Output: $\mathbf{a}$

1. Start with $\mathbf{a} = \phi$.
2. Do for each $l = (i, j)$ and $l \notin \mathbf{a}$:
   Compute $gain(\mathbf{a}, l)$
3. Terminate if $\forall l, gain(\mathbf{a}, l) \leq 1$.
4. Add the link $\hat{l}$ with the maximal $gain(\mathbf{a}, l)$ to $\mathbf{a}$.
5. Goto 2.

Greedy Vs. Hill climbing

	Greedy	Hill climbing
Operators	Add (a special case of Move)	Move Swap
Initial alignment	empty alignment	Viterbi alignment of a simple model
Applicability	log-linear models	fertility-based models
Algorithm type	greedy	greedy

Problems with the Search Algorithm

$$h(\mathbf{a},\mathbf{e},\mathbf{f}) = Pr(f_1^J, a_1^J | e_1^I)$$

$$= \binom{m-\phi_0}{\phi_0} {p_0}^{m-2\phi_0} {p_1}^{\phi_0} \prod_{i=1}^{l} \phi_i! n(\phi_i|e_i) \times$$

$$\prod_{j=1}^{m} t(f_j|e_{a_j}) d(j|a_j, l, m)$$

However, the search algorithm, which is general enough for any log-linear models, is not efficient for our models. It is time-consuming for each feature to figure out a probability when adding a new link, especially when the sentences are very long.

New Gain

$$gain(\mathrm{a}, l) = \sum_{m=1}^{M} l_m \log\left(\frac{h_m(\mathrm{a} \cup l, \mathrm{e}, \mathrm{f})}{h_m(\mathrm{a}, \mathrm{e}, \mathrm{f})} \right)$$

We restrict that $h_m(\mathrm{a,e,f}) \geq 0$ for all feature functions. Note that we still call the new heuristic function *gain* to reduce notational overhead. As a result, the termination condition will change to:

$$\sum_{m=1}^{M} l_m \log\left(\frac{h_m(\mathrm{a} \cup l, \mathrm{e}, \mathrm{f})}{h_m(\mathrm{a}, \mathrm{e}, \mathrm{f})} \right) \leq t$$

$$t = \sum_{m=1}^{M} l_m \left\{ \log\left(\frac{h_m(\mathrm{a} \cup l, \mathrm{e}, \mathrm{f})}{h_m(\mathrm{a}, \mathrm{e}, \mathrm{f})} \right) - \left[h_m(\mathrm{a} \cup l, \mathrm{e}, \mathrm{f}) - h_m(\mathrm{a}, \mathrm{e}, \mathrm{f}) \right] \right\}$$

We call *t* the gain threshold. It depends on the added link. But we remove this dependency for simplicity when using it in search algorithm by treating it as a fixed real-valued number.

Why we develop a new Gain?

- In the old *gain*, every feature has to figure out a probability; in the new *gain*, many terms will be cancelled out. For example, if a new link l=(i, j) is added, for IBM model 3 alone the new *gain* will only compute:

$$\frac{p_0 \times p_0}{p_1} \times \frac{f_0 \times (m - f_0 + 1)}{(m - 2f_0 + 1) \times (m - 2f_0 + 2)} \times (f_i + 1) \times$$

$$\frac{n(f_i + 1 \mid e_i)}{n(f_i \mid e_i)} \times \frac{t(f_j \mid e_i)}{t(f_j \mid e_0)} \times d(j \mid i, l, m)$$

The reason why we develop a new way to compute *gain* is that we try to reduce the computation.

New Search Algorithm

Input: $\mathbf{e}, \mathbf{f}, \mathbf{eT}, \mathbf{fT}, \mathbf{D}$ and t

Output: $\mathbf{a}$

1. Start with $\mathbf{a} = \phi$.
2. Do for each $l = (i, j)$ and $l \notin \mathbf{a}$:

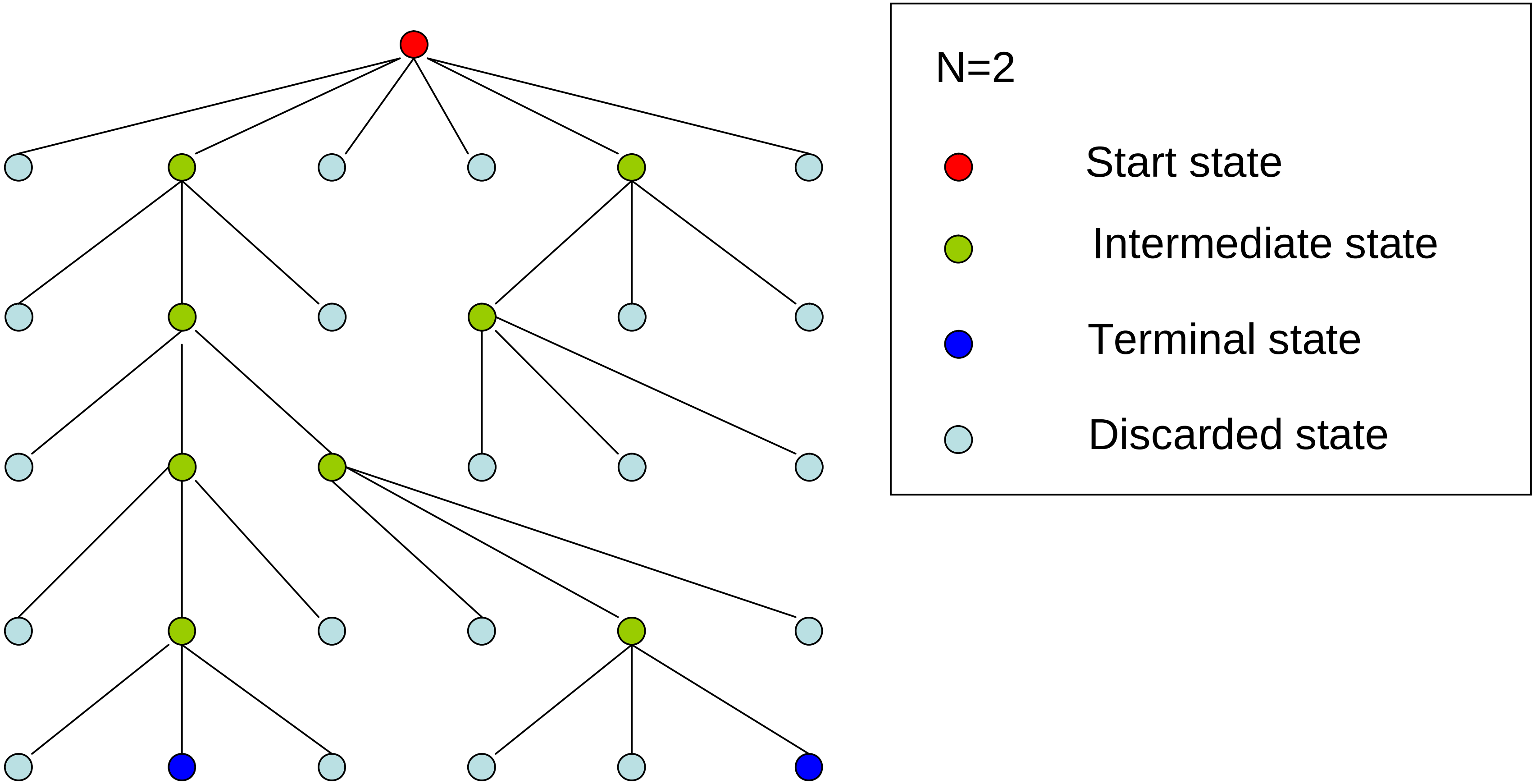
   Compute $gain(\mathbf{a}, l)$

3. Terminate if $\forall l, gain(\mathbf{a}, l) \leq t$.
4. Add the link $\hat{l}$ with the maximal $gain(\mathbf{a}, l)$ to $\mathbf{a}$.
5. Goto 2.

How to get a n-best list?

- As shown above, we use a greedy search algorithm to search the optimal alignment. When we use GIS algorithm to train model scaling parameters, we need a n-best list. Therefore, we use a breadth-first search algorithm with pruning. During the search, every link will be added to every alignment in the n-best list and then update the n-best list.
- During the search, states those are indistinguish will be recombined

An illustration



Experimental Results

		Chinese	English
Train	Sentences	108 925	
	Words	3 784 106	3 862 637
	Vocabulary	49 962	55 698
Dict	Entries	415 753	
	Vocabulary	206 616	203 497
Dev	Sentences	435	
	Words	11 462	14 252
	Ave. SentLen	26.35	32.76
Test	Sentences	500	
	Words	13 891	15 291
	Ave. SentLen	27.78	30.58

Statistics of training corpus (Train), bilingual dictionary (Dict), development corpus (Dev) and test corpus (Test)

Experimental Results (cont.)

	Size of Training Corpus				
	1K	5K	9K	39K	109K
Model 3 E $\rightarrow$ C	0.4497	0.4081	0.4009	0.3791	0.3745
Model 3 C $\rightarrow$ E	0.4688	0.4261	0.4221	0.3856	0.3469
Intersection	0.4588	0.4106	0.4044	0.3823	0.3687
Union	0.4596	0.4210	0.4157	0.3824	0.3703
Refined Method	0.4154	0.3586	0.3499	0.3153	0.3068
Model 3 E $\rightarrow$ C	0.4490	0.3987	0.3834	0.3639	0.3533
+ Model 3 C $\rightarrow$ E	0.3970	0.3317	0.3217	0.2949	0.2850
+ POS E $\rightarrow$ C	0.3828	0.3182	0.3082	0.2838	0.2739
+ POS C $\rightarrow$ E	0.3795	0.3160	0.3032	0.2821	0.2726
+ Dict	0.3650	0.3092	0.2982	0.2738	0.2685

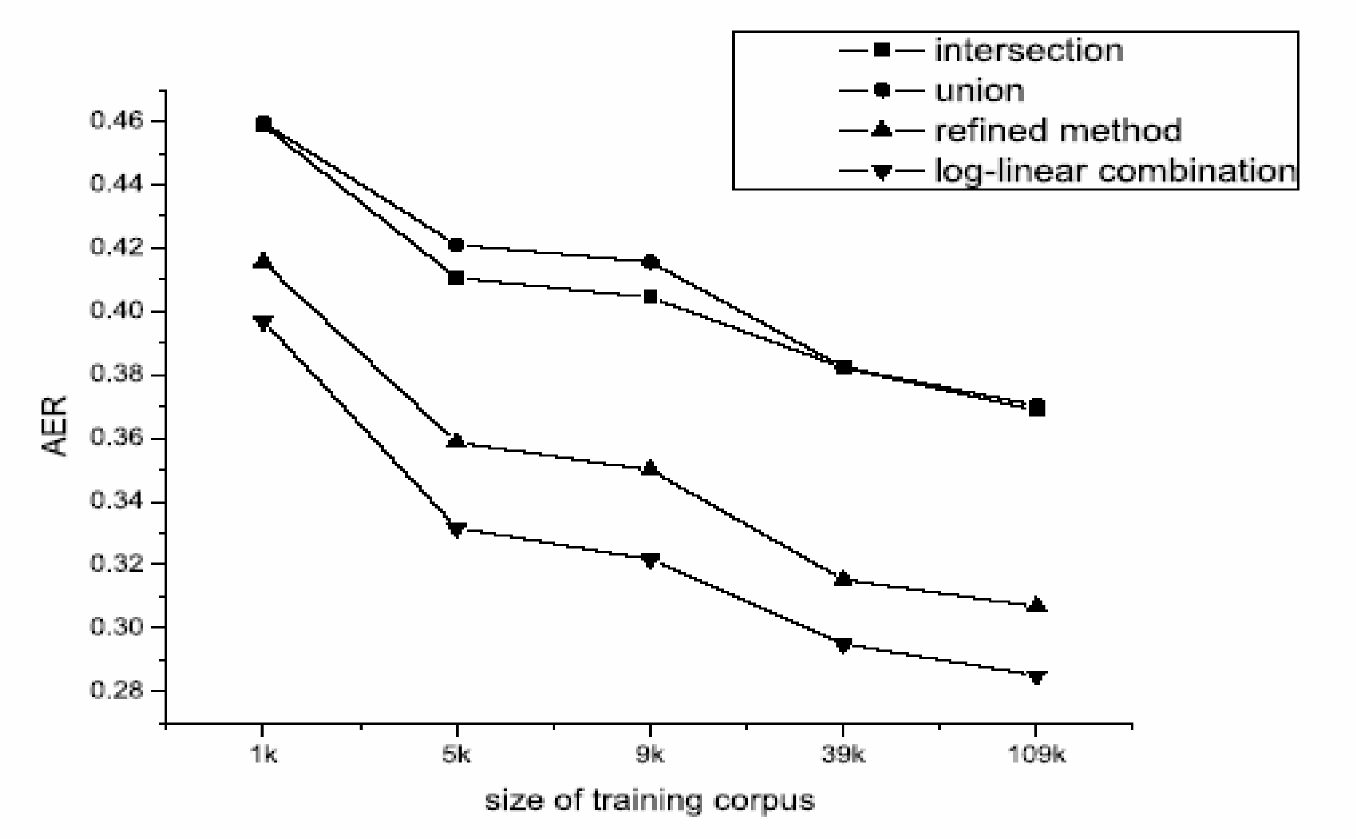
Comparison of AER for results of using IBM Model 3 (GIZA++) and log-linear models

Experimental Results (cont.)

	Size of Training Corpus				
	1K	5K	9K	39K	109K
Model 5 E → C	0.4384	0.3934	0.3853	0.3573	0.3429
Model 5 C → E	0.4564	0.4067	0.3900	0.3423	0.3239
Intersection	0.4432	0.3916	0.3798	0.3466	0.3267
Union	0.4499	0.4051	0.3923	0.3516	0.3375
Refined Method	0.4106	0.3446	0.3262	0.2878	0.2748
Model 3 E → C	0.4372	0.3873	0.3724	0.3456	0.3334
+ Model 3 C → E	0.3920	0.3269	0.3167	0.2842	0.2727
+ POS E → C	0.3807	0.3122	0.3039	0.2732	0.2667
+ POS C → E	0.3731	0.3091	0.3017	0.2722	0.2657
+ Dict	0.3612	0.3046	0.2943	0.2658	0.2625

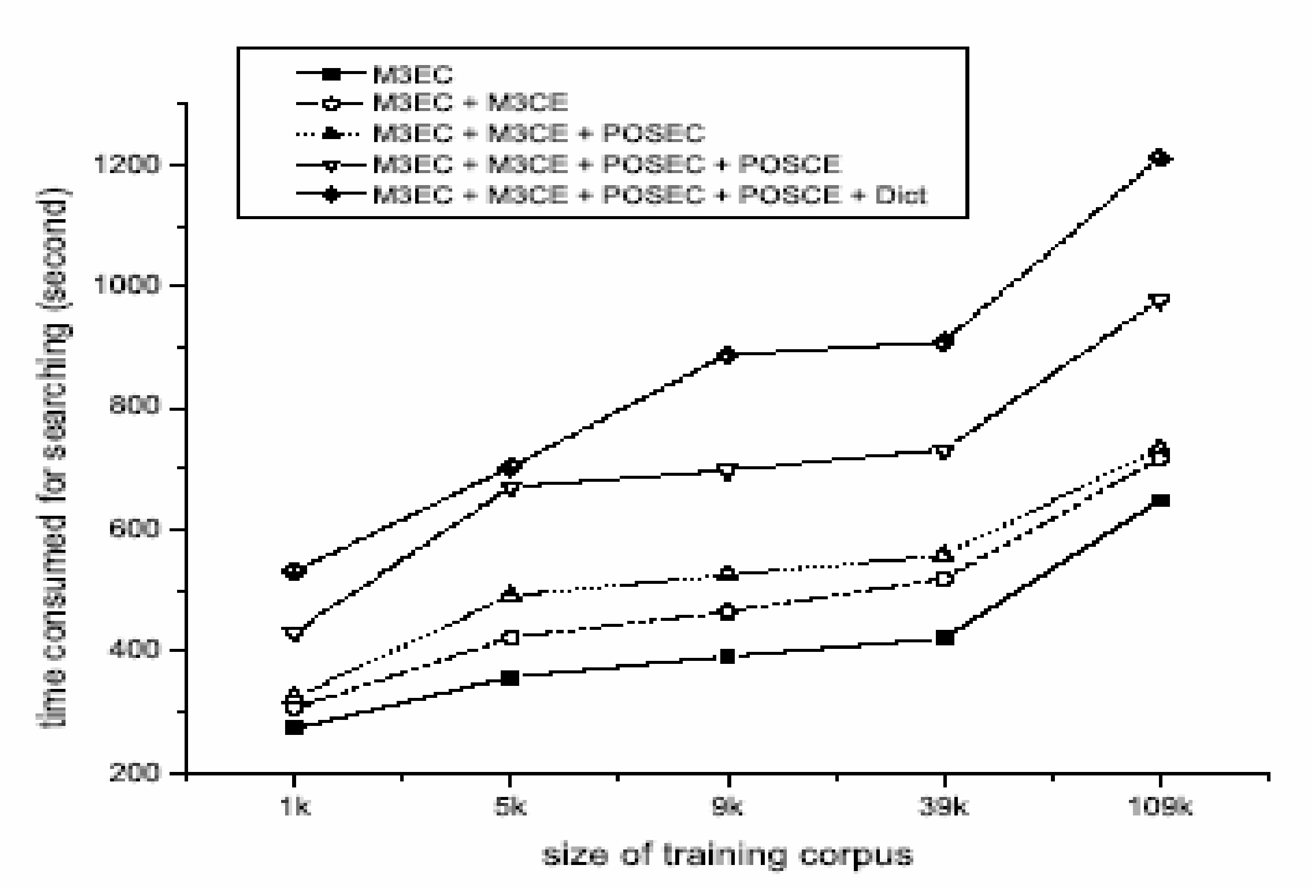
Comparison of AER for results of using IBM Model 5 (GIZA++) and log-linear models

Experimental Results (cont.)



Comparison on AER for various symmetrization methods: intersection, union, refined method and log-linear combination (i.e. M3 C->E + M3 E->C)

Experimental Results (cont.)



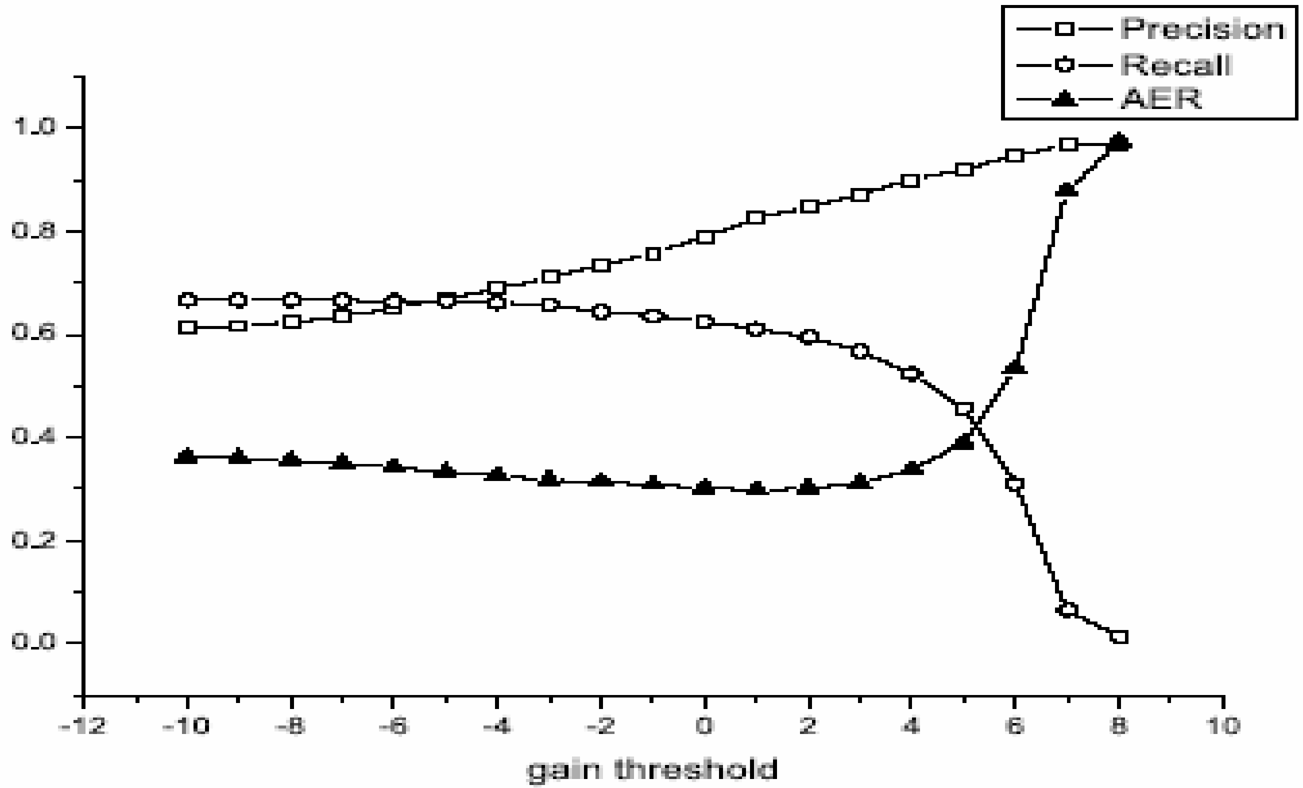
Effect of number of features and size of training corpus on search efficiency

Experimental Results (cont.)

	MEC	+MCE	+PEC	+PCE	+Dict
λ_1	1.000	0.466	0.291	0.202	0.151
λ_2	-	0.534	0.312	0.212	0.167
λ_3	-	-	0.397	0.270	0.257
λ_4	-	-	-	0.316	0.306
λ_5	-	-	-	-	0.119

Resulting model scaling factors

Experimental Results (cont.)



Precision, recall and AER over different gain thresholds with the same model scaling factors

Future Work

- Exploit more knowledge sources ranging from syntax-based models to various linguistic resources
- Optimize the model parameters directly with respect to AER
- Improve the efficiency of search algorithm
- Try on other language pairs

Thanks!