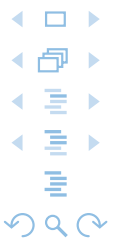
Statistical Machine Translation

Jia Xu

DFKI GmbH
Language Technology
D-66123 Saarbrücken
{Jia.Xu}@dfki.de

April 10, 2011

Overview



History of MT

- 1949 Shannon/Weaver: information theory
- 1952: Y. Bar-Hillel: research on MT at MIT
- 1954: Georgetown U.+IBM: Russian-English MT
- 1969: Chomsky: ban of statistics
  - ".. the notion of 'probability of a sentence' is an entirely useless one, under any known interpretation of this term."
- 1957: CalTec, Pasadena, CA;
- Peter Tom: Systran e.g. Babelfish system on AltaVista
- 1964: DFG, Germany

  ..
- 1990: Sato and Nagao: memory- and example-based translation
- 1989-1994: IBM Watson: Statistical Approach!

History of SMT

- 2003: Franz Och: Alignment template
- 2004: Philipp Koehn: Phrase based translation
- 2005: David Chiang: Hierarchical Phrase based translation
- 2006 - : ISI, ICT, etc.: Syntax based translation

Translation as A Classification Problem

- Data
  - Bilingual: parallel, comparable
  - Monolingual
- Training criterion
  - Maximum likelihood

$$\underset{\theta}{\operatorname{argmax}} \sum_{n=1}^{N} \log p_\theta(x_n|c_n)$$

  - Posterior probability

$$\underset{\theta}{\operatorname{argmax}} \sum_{n=1}^{N} \log p_\theta(c_n|x_n)$$

  - squared error criterion

$$\underset{\theta}{\operatorname{argmax}} \sum_{n=1}^{N} \sum_{c} [p_\theta(c|x_n) - \delta(c, c_n)]^2$$

Translation as A Classification Problem

- Probability Model
  - e.g. discriminant functions, NN, Gaussian, HMM, ME
  - ME: $p(c|x_1^D) = \frac{\prod_d \alpha_d(c|x_d)}{\sum_{c'} \prod_d \alpha_d(c'|x_d)}$
- Training algorithm, e.g.
  - EM: ML for hidden variables
  - Error back propagation: squared error criterion for NN
  - GIS (general iterative scaling): posterior probability for ME models
- Decision rule (search)
  - e.g. forward algorithm, dynamic programming, beam, A*
- Evaluation method
  - Accuracy: BLEU, NIST
  - Error rate: WER, PER, TER

Bayes Decision Rule

Bayes decision rule assuming probablistic dependencies between x and c:

- input: observation x
- output: decision c

$$
\begin{aligned}
x \rightarrow \hat{c} &= \underset{c}{\text{argmax}} \ \{Pr(c|x)\} \\
&= \underset{c}{\text{argmax}} \ \{Pr(c) \cdot Pr(x|c)\}
\end{aligned}
$$

- Question: why errors in addition to the minimum Bayes errors?

Bayes Approach for SMT

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Given:

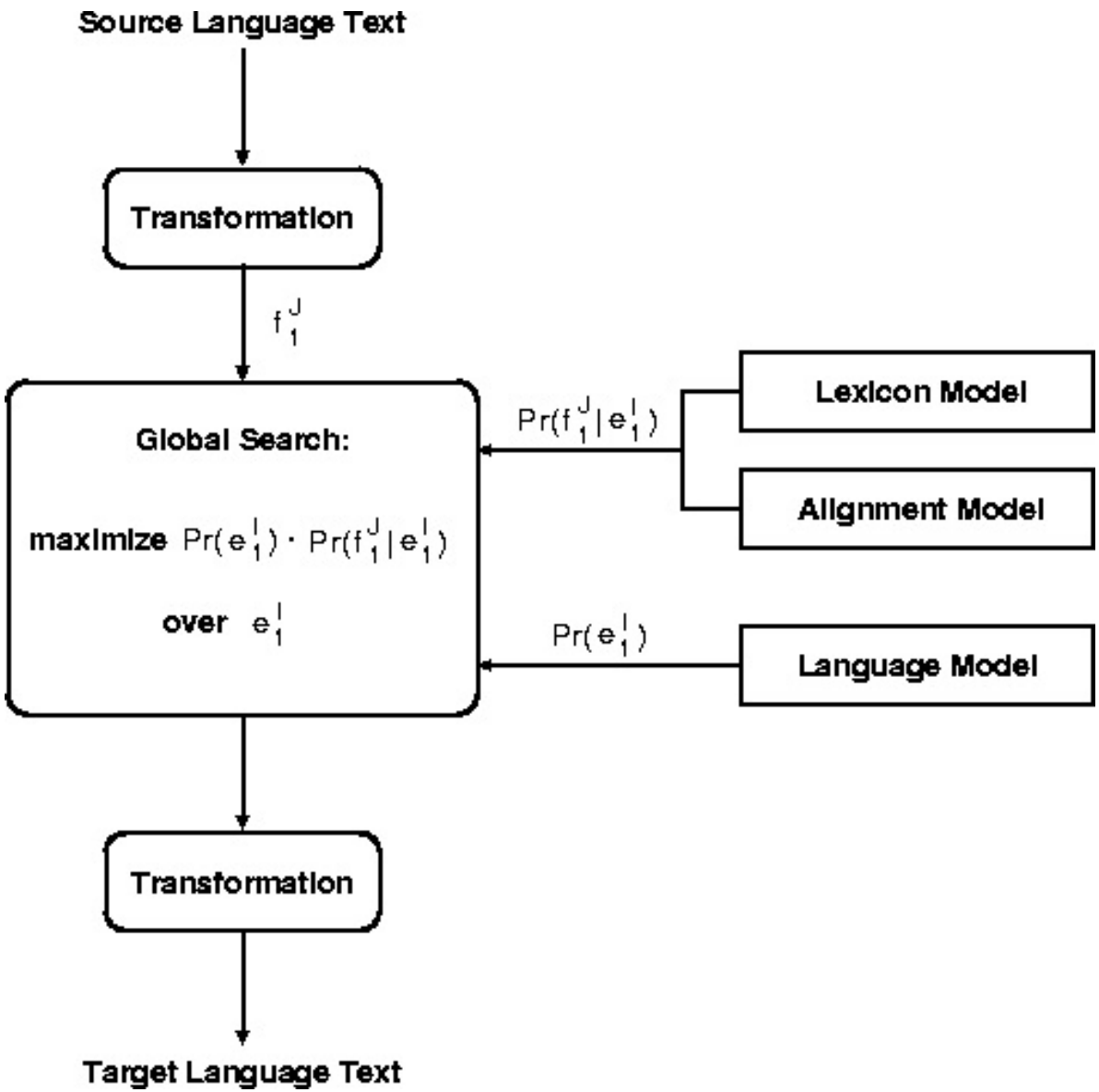
- a source sentence $f_1^J = f_1 \ldots f_j \ldots f_J$ to be translated into
- a target sentence $e_1^I = e_1 \ldots e_i \ldots e_I$

Choose the target sentence with the highest posterior probability:

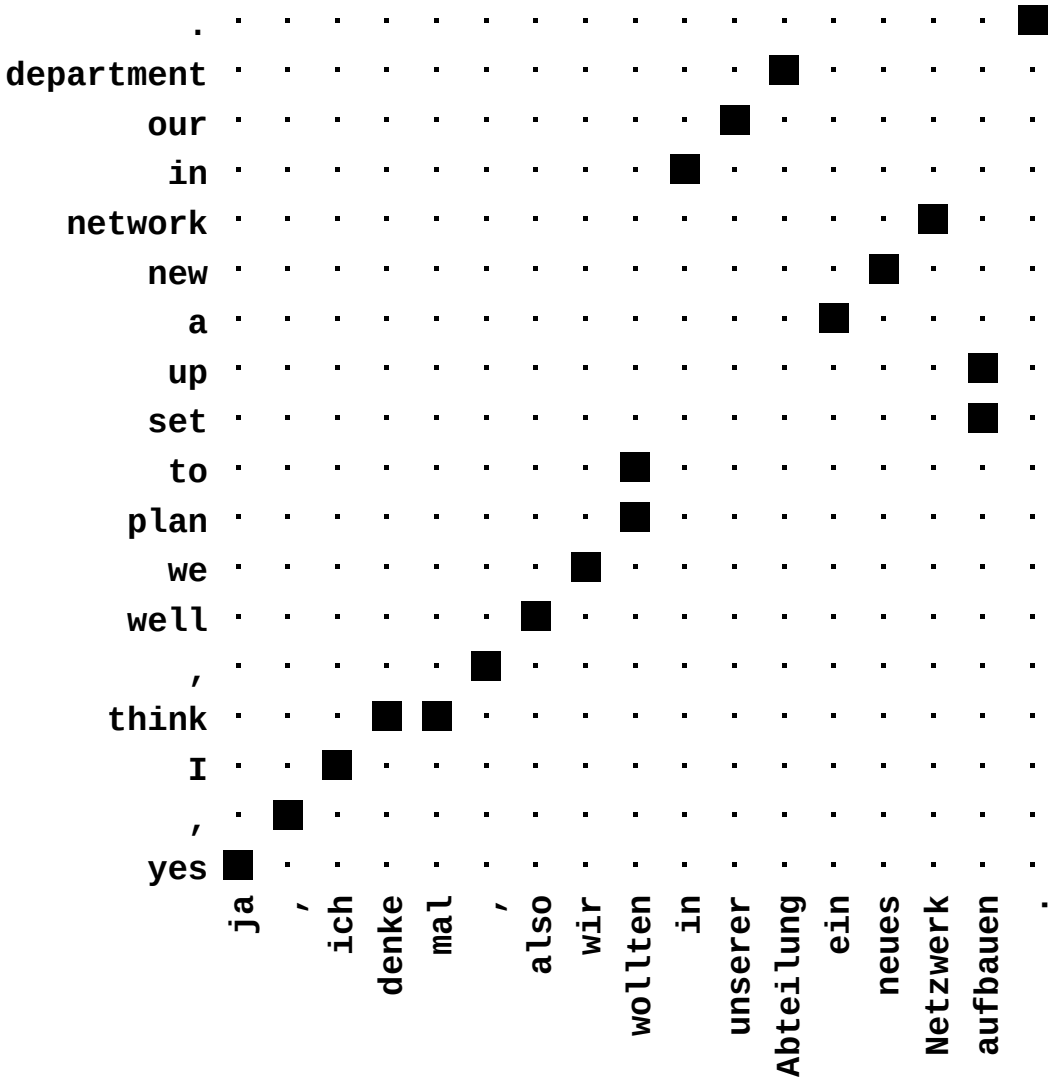
$$\begin{aligned} \hat{e}_1^I &= \underset{e_1^I}{\text{argmax}} \left\{ Pr(e_1^I | f_1^J) \right\} \\ &= \underset{e_1^I}{\text{argmax}} \left\{ Pr(e_1^I) \cdot Pr(f_1^J | e_1^I) \right\} \end{aligned}$$

- language model $Pr(e_1^I)$: well-formedness of target string
- translation model $Pr(f_1^J | e_1^I)$: translation relevance

Source-Channel Model



Word Alignment Model



A word alignment example from VerbMobil (black: Viterbi).

A Hidden Variable: Word Alignment

Introduce a 'hidden alignment' A

$$
\begin{aligned}
Pr(f_1^J|e_1^I) &= \sum_A Pr(f_1^J, A|e_1^I) \\
Pr(f_1^J, A|e_1^I) &= Pr(J|e_1^I) \cdot Pr(f_1^J, A|J, e_1^I) \\
&= Pr(J|e_1^I) \cdot Pr(A|J, e_1^I) \cdot Pr(f_1^J|A, J, e_1^I)
\end{aligned}
$$

- length probability: $Pr(J|e_1^I)$
- alignment probability: $Pr(A|J, e_1^I)$
- lexicon probability: $Pr(f_1^J|A, J, e_1^I)$

Restricted Word Alignment

$a_1^J \rightarrow A$

$$Pr(f_1^J|e_1^I) = \sum_{a_1^J} Pr(f_1^J, a_1^J|e_1^I)$$

$$Pr(f_1^J, a_1^J|e_1^I) = Pr(J|e_1^I) \cdot Pr(a_1^J|J, e_1^I) \cdot Pr(f_1^J|a_1^J, J, e_1^I)$$

- length probability:

$$Pr(J|e_1^I)$$

- alignment probability:

$$Pr(a_1^J|J, e_1^I) = \prod_{j=1}^{J} Pr(a_j|a_1^{j-1}, J, e_1^I)$$

- lexicon probability:

$$Pr(f_1^J|a_1^J, J, e_1^I) = \prod_{j=1}^{J} Pr(f_j|f_1^{j-1}, a_1^J, J, e_1^I)$$

Zero-Order Models: IBM model 1, 2

Do not consider dependency among alignments

Assumptions:

- length probability:

$$Pr(J|e_1^I) = P(J|I)$$

- alignment probability:

$$Pr(a_1^J|J, e_1^I) = \prod_{j=1}^{J} Pr(a_j|j, J, I)$$

- lexicon probability:

$$Pr(f_1^J|a_1^J, J, e_1^I) = \prod_{j=1}^{J} Pr(f_j|e_{a_j})$$

Zero-Order Models: IBM model 1, 2

In summary:

$$
\begin{aligned}
Pr(f_1^J|e_1^I) &= P(J|I) \cdot Pr(f_1^J|J, e_1^I) \\
&= P(J|I) \cdot \sum_{a_1^J} Pr(f_1^J, a_1^J|J, e_1^I) \\
&= P(J|I) \cdot \sum_{a_1^J} \prod_{j=1}^{J} P(a_j|j, J, I) \cdot P(f_j|e_{a_j})
\end{aligned}
$$

no interaction between different sums:

$$
Pr(f_1^J|e_1^I) = P(J|I) \cdot \prod_{j=1}^{J} \sum_{i=0}^{I} P(i|j, J, I) \cdot P(f_j|e_i)
$$

Zero-Order Models: IBM model 1, 2

For alignment model $P(i|j, J, I)$
IBM model 1:

- uniform probabilities
  $P(i|j, J, I) = \frac{1}{I+1}$
- important property: concave function of $p(f|e) \rightarrow$ one maximum, so any type of non-zero initialization is ok

IBM model 2:

- unconstrained probabilities
  $P(i|j, J, I)$: table of about $(J_{max} \cdot I_{max})^2$ entries

First-Order Model: HMM

$$
\begin{aligned}
Pr(f_1^J|e_1^I) &= P(J|I) \cdot Pr(f_1^J|J, e_1^I) \\
&= P(J|I) \cdot \sum_{a_1^J} Pr(f_1^J, a_1^I|J, e_1^I) \\
&= P(J|I) \cdot \sum_{a_1^J} \prod_{j=1}^{J} Pr(a_j|a_1^{j-1}, J, e_1^I) \cdot P(f_j|e_{a_j})
\end{aligned}
$$

For alignment model $Pr(a_j|a_1^{j-1}, J, e_1^I)$
HMM:

- consider alignment dependency on predecessor position a_{j-1}
  $Pr(a_j|a_1^{j-1}, J, e_1^I) = P(a_j|a_{j-1}, J, I)$

First-Order Model: HMM

$$
\begin{aligned}
Pr(f_1^J|e_1^I) &= P(J|I) \cdot Pr(f_1^J|J, e_1^I) \\
&= P(J|I) \cdot \sum_{a_1^J} Pr(f_1^J, a_1^I|J, e_1^I) \\
&= P(J|I) \cdot \sum_{a_1^J} \prod_{j=1}^{J} P(a_j|a_{j-1}, J, I) \cdot P(f_j|e_{a_j})
\end{aligned}
$$

For alignment model $Pr(a_j|a_1^{j-1}, J, e_1^I)$
HMM:

- consider alignment dependency on predecessor position a_{j-1}
  $Pr(a_j|a_1^{j-1}, J, e_1^I) = P(a_j|a_{j-1}, J, I)$

HMM: Forward Algorithm

Efficient calculation by Baum (forward) algorithm - Sum:

$$\sum_{a_1^J} \prod_{j=1}^{J} P(a_j|a_{j-1}, J, I) \cdot P(f_j|e_{a_j}) = \sum_{i=1}^{I} Q(i, J)$$

$$\begin{aligned} Q(i, j_0) &\doteq \sum_{a_1^{j_0}} \prod_{j=1}^{j_0} P(a_j|a_{j-1}, J, I) \cdot P(f_j|e_{a_j}) \\ &= \ldots \\ &= P(f_{j_0}|e_i) \cdot \sum_{i'} P(i|i', J, I) \cdot Q(i', j_0 - 1) \end{aligned}$$

HMM: Dynamic Programming

Efficient calculation with maximum approximation - Viterbi:

$$\sum_{a_1^J} \prod_{j=1}^{J} P(a_j|a_{j-1}, J, I) \cdot P(f_j|e_{a_j}) \approx \max_{i=1,\ldots,I} Q(i, J)$$

$$\begin{aligned} Q(i, j_0) &\doteq \max_{a_1^{j_0}} \prod_{j=1}^{j_0} P(a_j|a_{j-1}, J, I) \cdot P(f_j|e_{a_j}) \\ &= \ldots \\ &= P(f_{j_0}|e_i) \cdot \max_{i'} P(i|i', J, I) \cdot Q(i', j_0 - 1) \end{aligned}$$

HMM: Occupation Probability

$$\gamma(i|j_0) \doteq \frac{\sum_{a_1^J: a_{j_0}=i} P(a_1^J, f_1^J|J, e_1^I)}{\sum_{i'} \sum_{a_1^J: a_{j_0}=i'} P(a_1^J, f_1^J|J, e_1^I)} = \frac{Q(i, j_0) \cdot \tilde{Q}(i, j_0)}{\sum_{i'} Q(i', j_0) \cdot \tilde{Q}(i', j_0)}$$

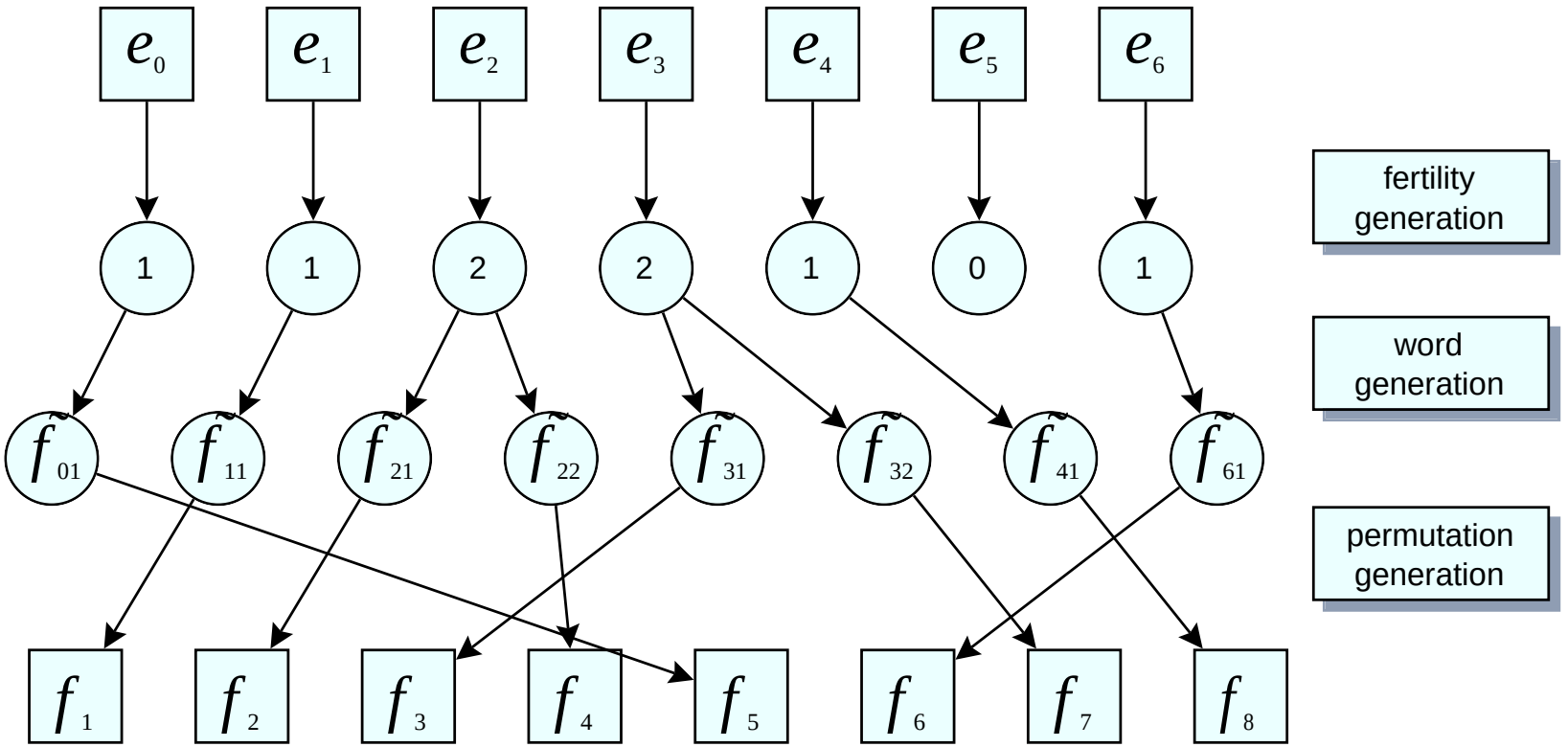
$$\sum_{a_1^J: a_{j_0}=i} P(a_1^J, f_1^J|J, e_1^I)$$

$$= \sum_{a_1^J: a_{j_0}=i} \prod_{j=1}^{J} P(a_j|a_{j-1}, J, I) P(f_j|e_{a_j})$$

$$= \sum_{a_1^{j_0}: a_{j_0}=i} \prod_{j=1}^{j_0} P(a_j|a_{j-1}, J, I) P(f_j|e_{a_j}) \sum_{a_{j_0+1}^J: a_{j_0}=i} \prod_{j=j_0+1}^{J} P(a_j|a_{j-1}, J, I) P(f_j|e_{a_j})$$

$$\doteq Q(i, j_0) \cdot \tilde{Q}(i, j_0)$$

Fertility Generation



Fertility based Models: IBM model 3, 4, 5

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

For alignment model $P(f_1^J, a_1^J | e_1^I)$
introduce fertility $\varphi(e)$: how many words does e generate, where

$$J = \sum_{i=0}^{I} \varphi(e_i)$$

IBM model 3:

- IBM model 2 + fertility

IBM model 4:

- dependence on jump width (permutation)
- dependence on word classes

IBM model 5:

- IBM model 4 is deficient; normalized version

Fertility Model: definition

$$Pr(f_1^J, a_1^J | e_1^I) = \sum_{(\tilde{f},\pi)\in(f_1^J,a_1^J)} Pr(\tilde{f}, \pi | e_1^I)$$

Generate (f_1^J, a_1^J)

1. select fertility for each e_i:

$$\varphi_i \doteq \varphi(e_i) = \sum_{j=1}^{J} \delta(i, a_j)$$

2. for each e_i, select a tablet of French words:

$$\tilde{f}_{iv}, v = 1, ..., \varphi(e_i) \text{ so that } a_{\pi_{iv}} = i$$

3. select a permutation of the sequence $\tilde{f}$:

$$\pi : (i, v) \rightarrow j = \pi_{iv} \text{ where } \tilde{f}_{iv} = f_{\pi_{iv}}$$

Fertility Model

$$Pr(\tilde{f}, \pi | e_0^I) = Pr(\varphi_0^I | e_0^I) \cdot Pr(\tilde{f} | \varphi_0^I, e_0^I) \cdot Pr(\pi | \tilde{f}, \varphi_0^I, e_0^I)$$

- empty position: $i = 0$

1. fertility generation (IBM model 3, 4, 5):

$$Pr(\varphi_0^I | e_0^I) = P(\varphi_0 | e_0, \sum_{i=1}^{I} \varphi_i) \cdot \prod_{i=1}^{I} P(\varphi_i | e_i)$$

2. word generation (IBM model 3, 4, 5):

$$Pr(\tilde{f} | \varphi_0^I, e_0^I) = \prod_{i=0}^{I} \prod_{v=1}^{\varphi_i} Pr(\tilde{f}_{iv} | e_i)$$

3. permutation (only IBM model 3):

$$Pr(\pi | \tilde{f}, \varphi_0^I, e_0^I) = \frac{1}{\varphi_0!} \cdot \prod_{i=1}^{I} \prod_{v=1}^{\varphi_i} P(\pi_{iv} | i, I, J)$$

Fertility Model: IBM model 3

$$
\begin{aligned}
Pr(f_1^J, a_1^J | e_0^I) &= \sum_{(\tilde{f},\pi)\in(f_1^J,e_1^J)} Pr(\tilde{f}, \pi | e_1^I) \\
&= \ldots \\
&= P(\varphi_0|e_0, \sum_{i=1}^{I}\varphi_i) \cdot \prod_{i=1}^{I}\varphi_i! Pr(\varphi_i|e_i) \\
&\quad \cdot \prod_{j=1}^{J} P(f_j|e_{a_j}) \cdot \prod_{j=1, a_j\neq 0}^{J} P(j|a_j, I, J)
\end{aligned}
$$

in above sum, there are $\prod_{i=0}^{I}\varphi_i!$ equally probable terms.

Fertility Model: IBM model 4

Detailed permutation model:

$$Pr(\pi|\tilde{f}, \varphi_0^I, e_0^I) = \frac{1}{\varphi_0!} \cdot \prod_{i=1}^{I} \prod_{v=1}^{\varphi_i} P(\pi_{iv}|\cdots)$$

First-order dependence along j-axis with homogeneous probabilities ($j = \pi_{iv}$):

$$P(\pi_{iv}|..) \doteq \begin{cases} P_{=1}(\pi_{iv} - \tilde{\pi}_{<i-1>}|G(f_{\pi_{iv}}), G(e_{<i-1>})) & : v = 1 \\ P_{>1}(\pi_{iv} - \pi_{i,v-1}|G(f_{\pi_{iv}})) & : v > 1 \wedge \pi_{iv} > \pi_{i,v-1} \\ 0 & : v > 1 \wedge \pi_{iv} \leq \pi_{i,v-1} \end{cases}$$

with

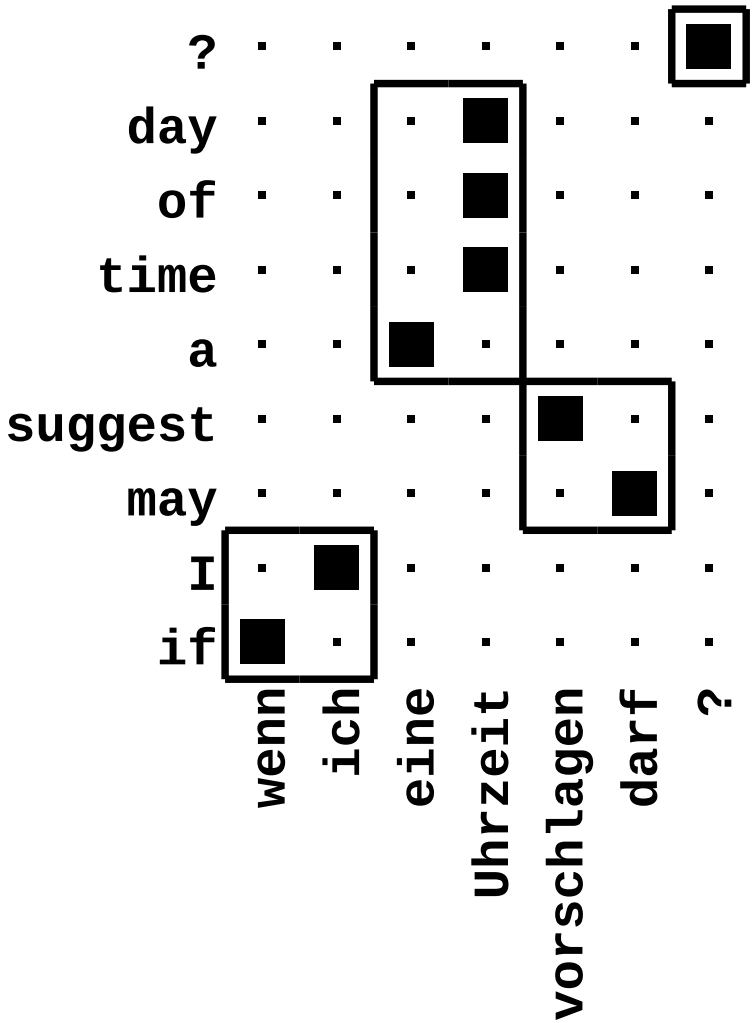
$$\tilde{\pi}_i \doteq \text{int}(.5 + \frac{1}{\varphi_i} \sum_{v=1}^{\varphi_i} \pi_{iv})$$

$$< i > \doteq \max_{i' \leq i} \{i' : \varphi_{i'} > 0\}$$

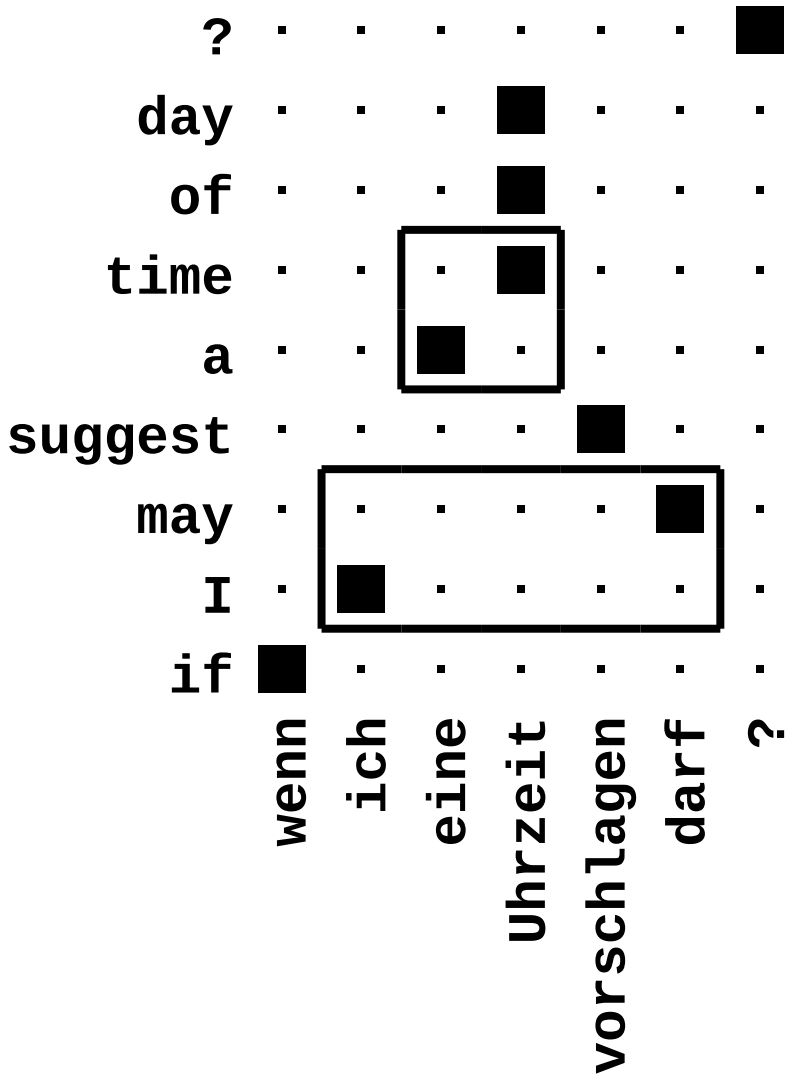
$$G(e), G(f) \doteq \text{word classes from clustering}$$

Phrase Extraction: Example

consistent phrase pairs:

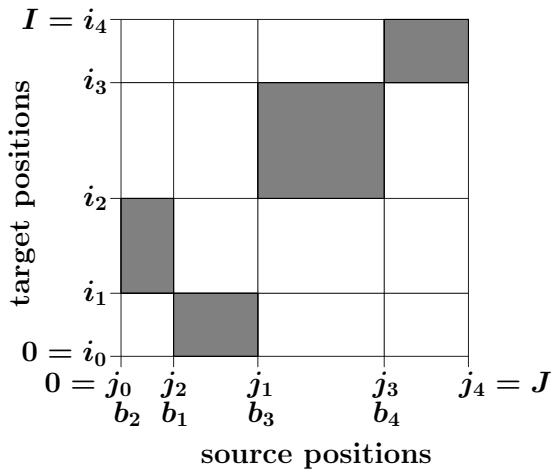


inconsistent phrase pairs:



Phrase Segmentation

- segmentation of (f_1^J, e_1^I):
  $k \rightarrow s_k := (i_k; b_k, j_k),$ for $k = 1 \ldots K$
- i_k: last position of the k^{th} target phrase; we set $i_0 := 0$
- (b_k, j_k): start and end positions of the source phrase that is aligned to the k^{th} target phrase; we set $j_0 := 0$.
- $\tilde{e}_k := e_{i_{k-1}+1} \ldots e_{i_k}$ and $\tilde{f}_k := f_{b_k} \ldots f_{j_k}$



Phrase Scoring

$$
\begin{aligned}
Q(f_1^J, e_1^I; s_1^K) \quad = \quad & \prod_{i=1}^{I} [c_1 \cdot P(e_i|e_{i-1}^{i-1})^{\lambda_1}] \\
& \cdot \prod_{k=1}^{K} [c_2 \cdot P(\tilde{f}_k|\tilde{e}_k)^{\lambda_2} \cdot P(\tilde{e}_k|\tilde{f}_k)^{\lambda_3} \\
& \prod_{j=j_{k-1}+1}^{j_k} P(f_j|\tilde{e}_k)^{\lambda_4} \cdot \prod_{i=i_{k-1}+1}^{i_k} P(e_i|\tilde{f}_k)^{\lambda_5}
\end{aligned}
$$

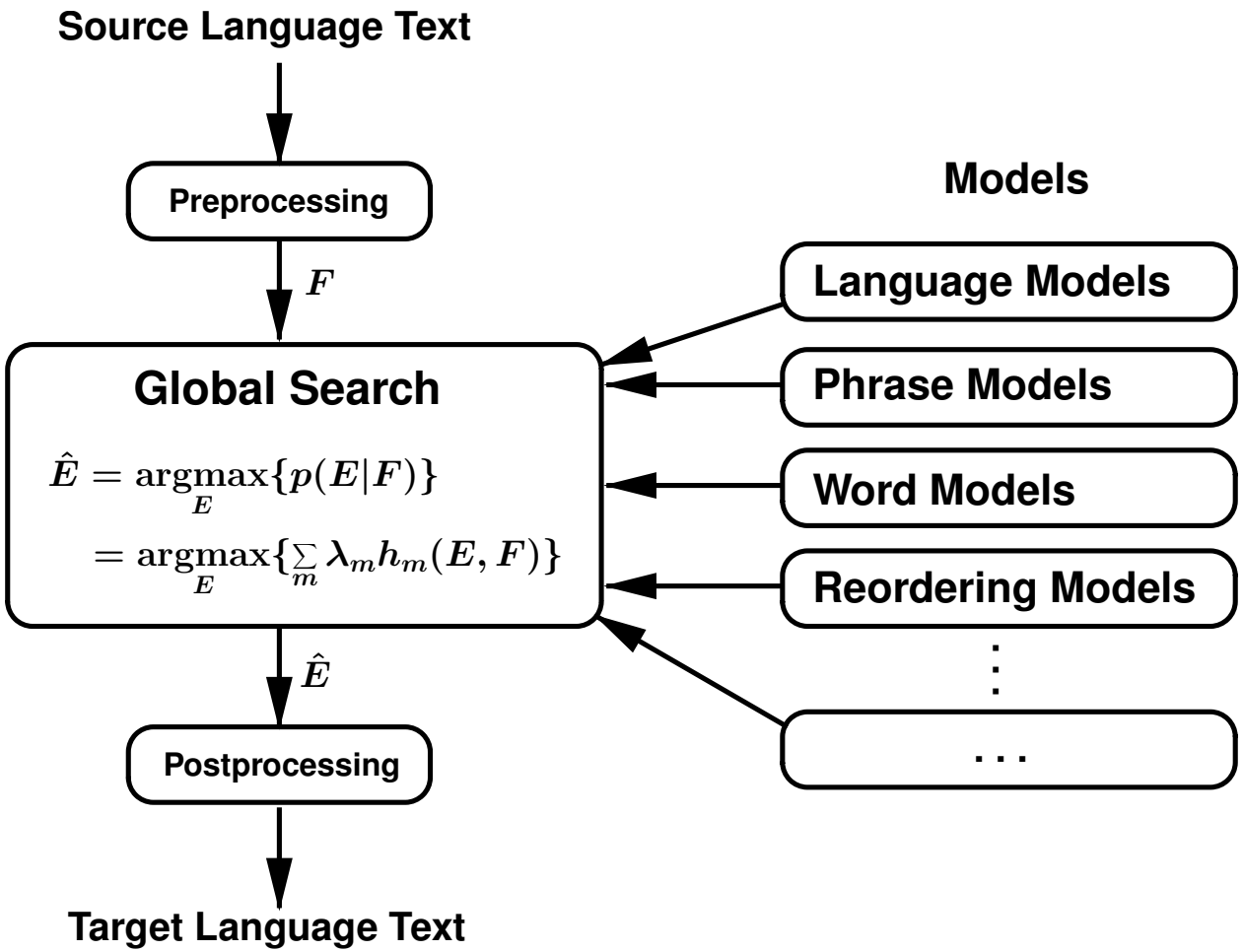
Translation Model

- various types of dependencies
- combination using log-linear framework:

$$p(E|F) = \frac{\exp\left[\sum_m \lambda_m h_m(E,F)\right]}{\sum_{\tilde{E}} \exp\left[\sum_m \lambda_m h_m(\tilde{E},F)\right]}$$

  with models (feature functions) $h_m(E,F), m = 1, ..., M$
- Bayes decision rule:

$$\begin{aligned} F \rightarrow \hat{E}(F) &= \underset{E}{\operatorname{argmax}} \left\{ p(E|F) \right\} \\ &= \underset{E}{\operatorname{argmax}} \left\{ \sum_m \lambda_m h_m(E,F) \right\} \end{aligned}$$


Source Language Text
Preprocessing
F
Global Search
$\hat{E} = \underset{E}{\operatorname{argmax}}\{p(E|F)\}$
$= \underset{E}{\operatorname{argmax}}\{\sum_m \lambda_m h_m(E,F)\}$
$\hat{E}$
Postprocessing
Target Language Text
Models
Language Models
Phrase Models
Word Models
Reordering Models
...

Models

Baseline:

- phrase lexica $p(f_1^J|e_1^I)$ and $p(e_1^I|f_1^J)$
- single-word lexica $p(f|e)$ and $p(e|f)$
- n-gram language model (LM)
- word and phrase penalty
- distance-based penalty for reordering

Hierarchical Phrase based Translation

- models uses a grammar consisting of SCFG (Synchronous Context-Free Grammar) rules.
  - ne X1 pas $\rightarrow$ not X1 (French-English)
  - ate X1 $\rightarrow$ habe X1 gegessen (English-German)
  - X1 of the X2 $\rightarrow$ le X2 X1 (English-French)
- Input: Das Tor geht schnell auf
- Rules:
  - Das Tor $\rightarrow$ The door
  - schnell $\rightarrow$ quickly
  - geht X1 auf $\rightarrow$ opens X1
  - X1 X2 $\rightarrow$ X1 X2
- Output: Tree structure mapping

References:

- Hermann Ney. Natural Language processing: Statistical Methods. RWTH-Aachen Lecture Notes. 2004. Aachen, Germany
- Duda and Hart and Stork. Pattern Classification. 2000