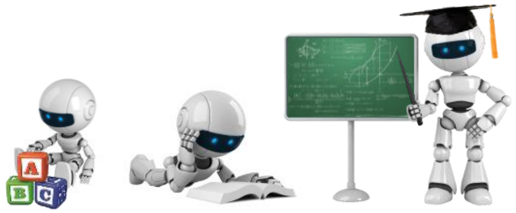
USC Viterbi
School of Engineering
Department of Computer Science

CSCI 544
Applied Natural Language Processing

Mohammad Rostami
USC Computer Science Department

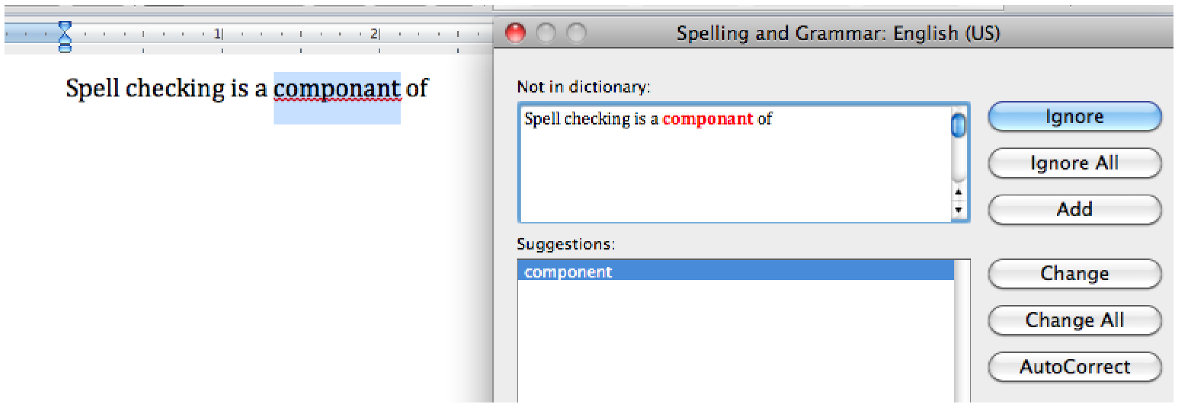
Slides adopted from slides by Jurfsky, Martin, and Collins



Spelling Correction

- Applications

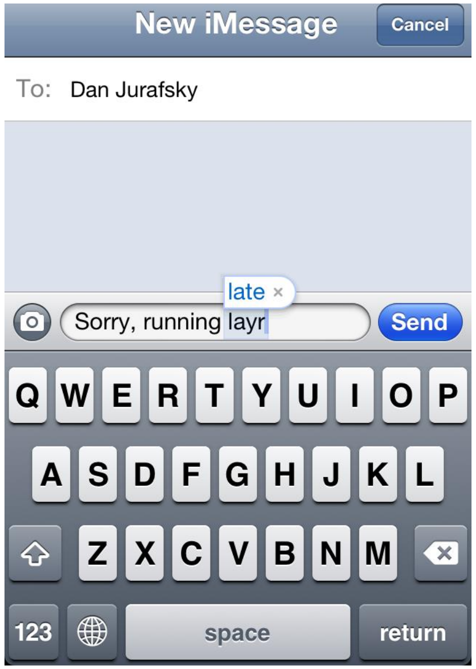
Word processing



Spell checking is a componant of

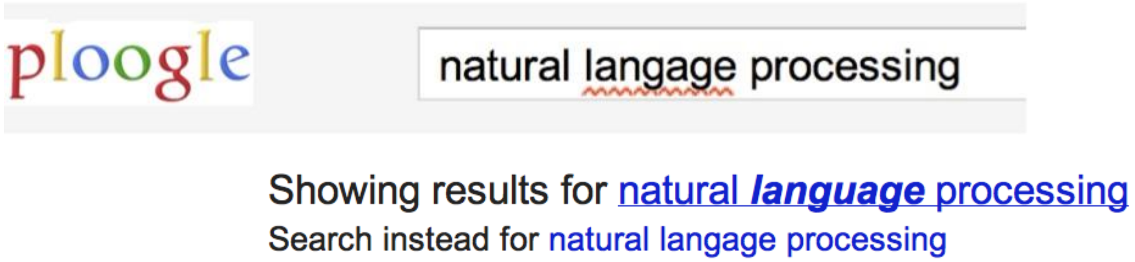
Spell checking is a componant of

Phones



Sorry, running layr

Web search



natural langage processing

Showing results for natural *language* processing

Search instead for natural langage processing

Spelling Correction

- How common are spelling errors:

- **26**%: Web queries Wang *et al.* 2003
- **13**%: Retyping, no backspace: Whitelaw *et al.* English&German
- **7**%: Words corrected retyping on phone-sized organizer
- **1-2**%: Retyping: Kane and Wobbrock 2007, Gruden et al. 1983

Spelling Correction

- Spelling Tasks:

- Spelling Error Detection
- Spelling Error Correction:
  - Autocorrect
    hte→the
  - Suggest a correction
  - Suggestion lists

- Spelling error types:

- Non-word Errors
  - *graffe* →*giraffe*
- Real-word Errors
  - Typographical errors
    - *three* →*there*
  - Cognitive Errors
    piece→*peace*,
    - *too* → *two*

Non-word Spelling Errors

- Non-word spelling error detection:
  - Any word not in a ***dictionary*** is an error
  - The larger the dictionary the better
- Non-word spelling error correction:
  - Generate ***candidates***: real words that are similar to error
  - Choose the one which is best: noisy channel model

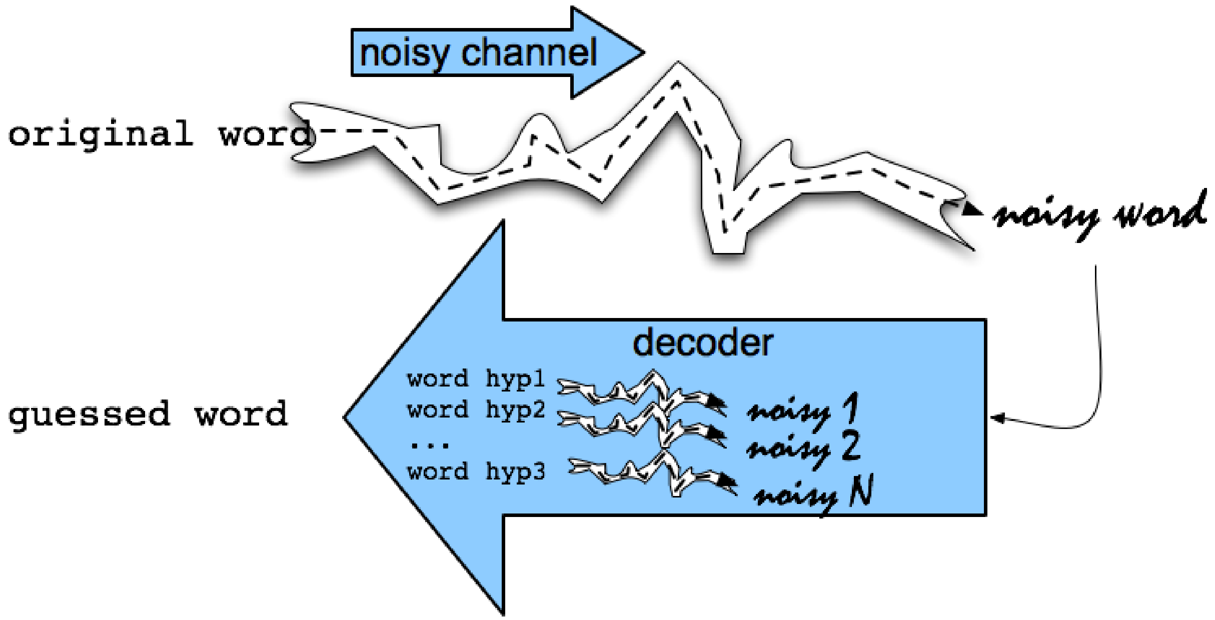
Real Word Spelling Errors

- For each word *w*, generate candidate set:
  - Find candidate words with similar ***pronunciations***
  - Find candidate words with similar ***spelling***
  - Include *w* in candidate set
- Choose best candidate
  - Noisy Channel

Spelling Correction: Noisy Channel Model

- We see an observation x of a misspelled word
- Find the correct word w

$$\hat{w} = \underset{w \in V}{\operatorname{argmax}} P(w \mid x) \qquad = \underset{w \in V}{\operatorname{argmax}} \frac{P(x \mid w)P(w)}{P(x)} \qquad = \underset{w \in V}{\operatorname{argmax}} \underbrace{P(x \mid w)}_{\text{Channel}}\underbrace{P(w)}_{\text{Language Model: N-Gram}}$$



Spelling Correction

- Example:

Kernighan, Mark D., Kenneth W. Church, and William A. Gale. 1990. A spelling correction program based on a noisy channel model. Proceedings of COLING 1990, 205-210

- acress

- Candidate Generation:

  - Words with similar spelling

- Damerau-Levenshtein edit distance:

  – Insertion: cress
  – Deletion: actress
  – Substitution: caress
  – Transposition of two adjacent letters: arcoss

Candidate Generation

- 80% of errors are within edit distance 1
- Almost all errors within edit distance 2
- Also allow insertion of **space** or **hyphen**
  - `thisidea` → `this idea`
  - `inlaw` → `in-law`

Candidate Generation

- Words within 1 edit of acress

Error	Candidate Correction	Correct Letter	Error Letter	Type
acress	actress	t	-	deletion
acress	cress	-	a	insertion
acress	caress	ca	ac	transposition
acress	access	c	r	substitution
acress	across	o	e	substitution
acress	acres	-	s	insertion

Language Model

word	Frequency of word	P(word)
actress	9,321	.0000230573
cress	220	.0000005442
caress	686	.0000016969
access	37,038	.0000916207
across	120,844	.0002989314
acres	12,874	.0000318463

Channel Model

- Edit probability: $P(x|w)$
  - *Kernighan, Church, Gale 1990*

- *Misspelled word* $x = x_1, x_2, x_3 \ldots x_m$
- *Correct word* $w = w_1, w_2, w_3, \ldots, w_n$

- Spelling errors tend to be character level

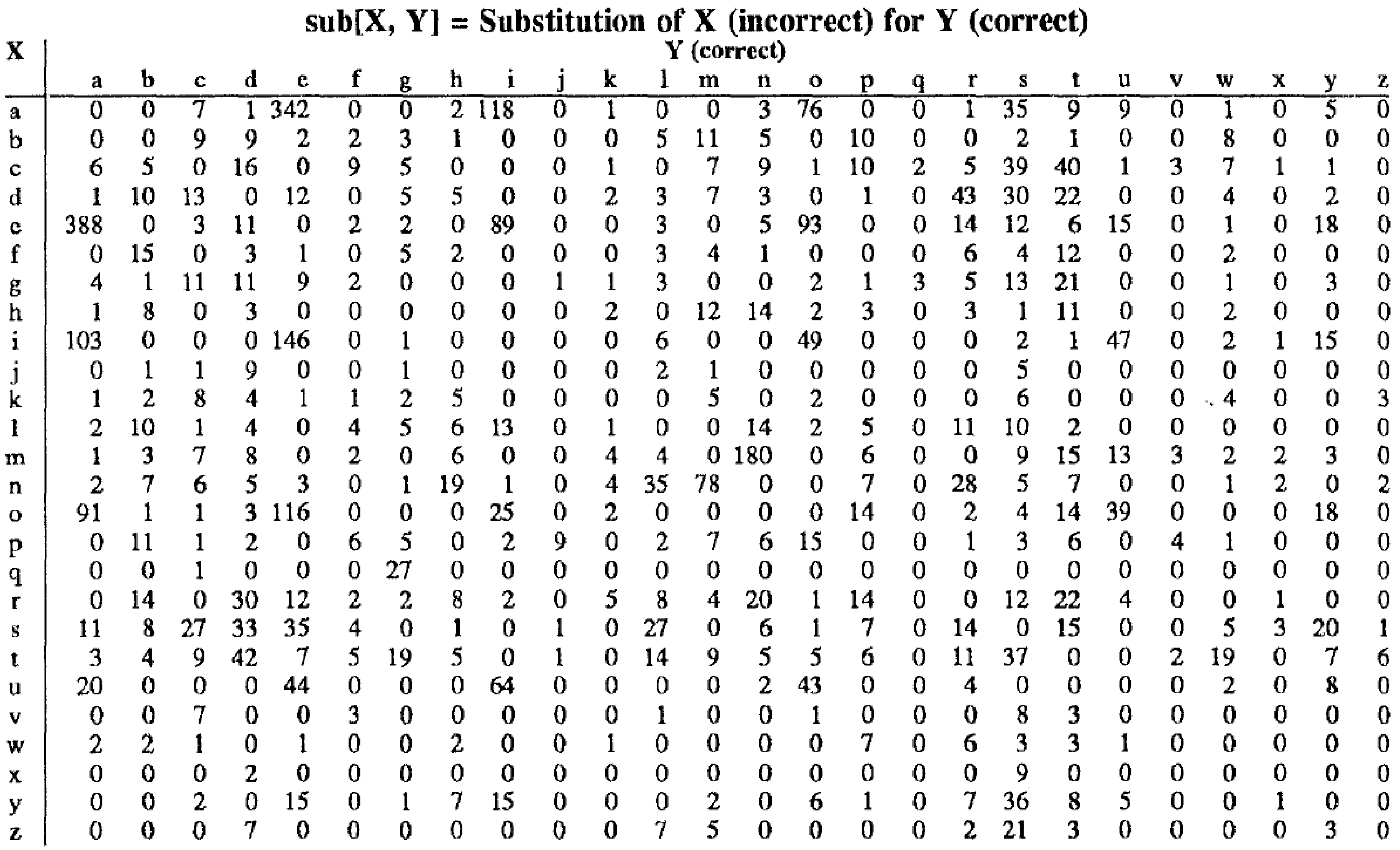
- studeid, freind
- pronuon, ruote

Channel Model

- Edit probability: P(x|w)
  - *Kernighan, Church, Gale 1990*
- *Misspelled word* $x = x_1, x_2, x_3 \ldots x_m$
- *Correct word* $w = w_1, w_2, w_3, \ldots, w_n$
  - (deletion/insertion/substitution/transposition)
  - Insertion and deletion conditioned on previous character

```
del[x,y]:     count(xy typed as x)
ins[x,y]:     count(x typed as xy)
sub[x,y]:     count(x typed as y)
trans[x,y]:   count(xy typed as yx)
```

Confusion Matrix



sub[X, Y] = Substitution of X (incorrect) for Y (correct)

X	Y (correct)																									
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	0	7	1	342	0	0	2	118	0	1	0	0	3	76	0	0	1	35	9	9	0	1	0	5	0
b	0	0	9	9	2	2	3	1	0	0	0	5	11	5	0	10	0	0	2	1	0	0	8	0	0	0
c	6	5	0	16	0	9	5	0	0	0	1	0	7	9	1	10	2	5	39	40	1	3	7	1	1	0
d	1	10	13	0	12	0	5	5	0	0	2	3	7	3	0	1	0	43	30	22	0	0	4	0	2	0
e	388	0	3	11	0	2	2	0	89	0	0	3	0	5	93	0	0	14	12	6	15	0	1	0	18	0
f	0	15	0	3	1	0	5	2	0	0	0	3	4	1	0	0	0	6	4	12	0	0	2	0	0	0
g	4	1	11	11	9	2	0	0	0	1	1	3	0	0	2	1	3	5	13	21	0	0	1	0	3	0
h	1	8	0	3	0	0	0	0	0	0	2	0	12	14	2	3	0	3	1	11	0	0	2	0	0	0
i	103	0	0	0	146	0	1	0	0	0	0	6	0	0	49	0	0	0	2	1	47	0	2	1	15	0
j	0	1	1	9	0	0	1	0	0	0	0	2	1	0	0	0	0	0	5	0	0	0	0	0	0	0
k	1	2	8	4	1	1	2	5	0	0	0	0	5	0	2	0	0	0	6	0	0	0	4	0	0	3
l	2	10	1	4	0	4	5	6	13	0	1	0	0	14	2	5	0	11	10	2	0	0	0	0	0	0
m	1	3	7	8	0	2	0	6	0	0	4	4	0	180	0	6	0	0	9	15	13	3	2	2	3	0
n	2	7	6	5	3	0	1	19	1	0	4	35	78	0	0	7	0	28	5	7	0	0	1	2	0	2
o	91	1	1	3	116	0	0	0	25	0	2	0	0	0	0	14	0	2	4	14	39	0	0	0	18	0
p	0	11	1	2	0	6	5	0	2	9	0	2	7	6	15	0	0	1	3	6	0	4	1	0	0	0
q	0	0	1	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	0	14	0	30	12	2	2	8	2	0	5	8	4	20	1	14	0	0	12	22	4	0	0	1	0	0
s	11	8	27	33	35	4	0	1	0	1	0	27	0	6	1	7	0	14	0	15	0	0	5	3	20	1
t	3	4	9	42	7	5	19	5	0	1	0	14	9	5	5	6	0	11	37	0	0	2	19	0	7	6
u	20	0	0	0	44	0	0	0	64	0	0	0	0	2	43	0	0	4	0	0	0	0	2	0	8	0
v	0	0	7	0	0	3	0	0	0	0	0	1	0	0	1	0	0	0	8	3	0	0	0	0	0	0
w	2	2	1	0	1	0	0	2	0	0	1	0	0	0	0	7	0	6	3	3	1	0	0	0	0	0
x	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0
y	0	0	2	0	15	0	1	7	15	0	0	0	2	0	6	1	0	7	36	8	5	0	0	1	0	0
z	0	0	0	7	0	0	0	0	0	0	0	7	5	0	0	0	0	2	21	3	0	0	0	0	3	0

Channel Model

$$P(x|w) = \begin{cases} \frac{\text{del}[w_{i-1},w_i]}{\text{count}[w_{i-1}w_i]}, & \text{if deletion} \\ \frac{\text{ins}[w_{i-1},x_i]}{\text{count}[w_{i-1}]}, & \text{if insertion} \\ \frac{\text{sub}[x_i,w_i]}{\text{count}[w_i]}, & \text{if substitution} \\ \frac{\text{trans}[w_i,w_{i+1}]}{\text{count}[w_i w_{i+1}]}, & \text{if transposition} \end{cases}$$

Kernighan, Church, Gale 1990

Noisy Channel Model for "acress"

Candidate Correction	Correct Letter	Error Letter	x|w	P(x|word)
actress	t	-	c|ct	.000117
cress	-	a	a|#	.00000144
caress	ca	ac	ac|ca	.00000164
access	c	r	r|c	.000000209
across	o	e	e|o	.0000093
acres	-	s	es|e	.0000321

Candidate Correction	Correct Letter	Error Letter	x|w	P(x|word)	P(word)	10^9 *P(x|w)P(w)
actress	t	-	c|ct	.000117	.0000231	2.7
cress	-	a	a|#	.00000144	.000000544	.00078
caress	ca	ac	ac|ca	.00000164	.00000170	.0028
access	c	r	r|c	.000000209	.0000916	.019
across	o	e	e|o	.0000093	.000299	2.8
acres	-	s	es|e	.0000321	.0000318	1.0

Using Bigram for “acress”

- “a stellar and versatile **acress** whose combination of sass and glamour…”
- Counts from the Corpus of Contemporary American English with add-1 smoothing
- P(actress|versatile)=.000021
  P(whose|actress) = .0010
- P(across|versatile) =.000021
  P(whose|across) = .000006

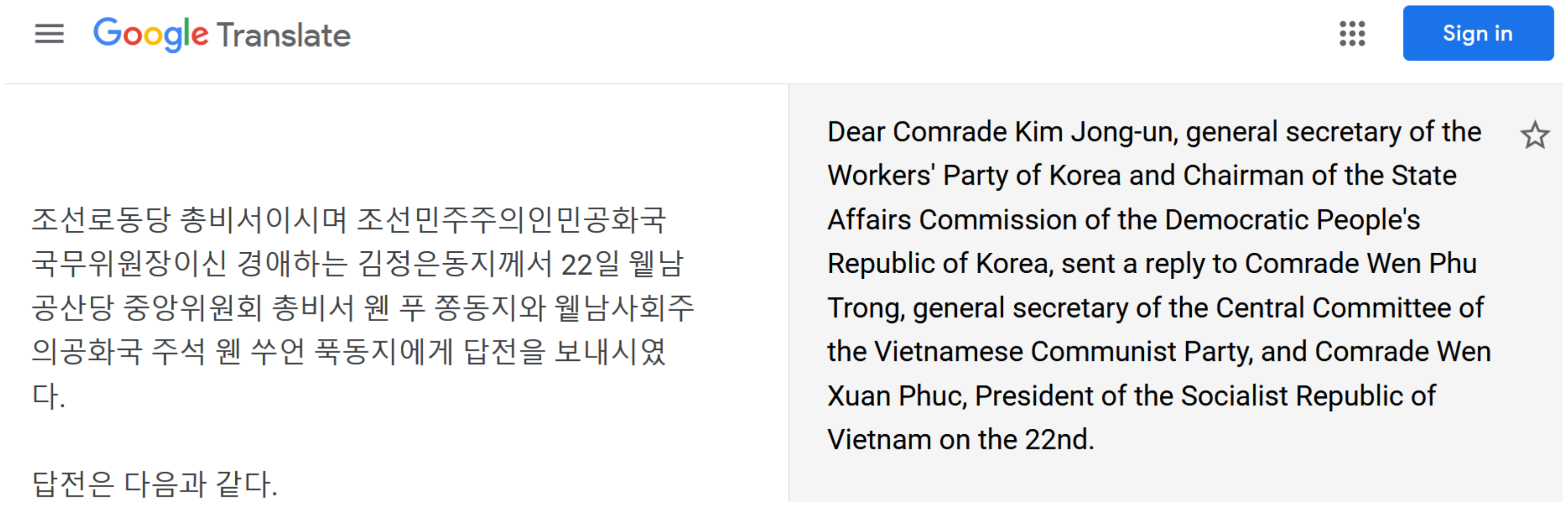
- P(“versatile actress whose”) = .000021*.0010 = 210 $\times 10^{-10}$
- P(“versatile across whose”) = .000021*.000006 = 1 $\times 10^{-10}$

Spelling Correction: Evaluation

- Some spelling error test sets
  - Wikipedia’s list of common English misspelling
  - Aspell filtered version of that list
  - Birkbeck spelling error corpus
  - Peter Norvig’s list of errors

Machine Translation

- **Translation: a very challenging task in general**
  - poetry
  - old text
  - professional text
- **Machine Translation**
  - Information access



조선로동당 총비서이시며 조선민주주의인민공화국 국무위원장이신 경애하는 김정은동지께서 22일 웰남 공산당 중앙위원회 총비서 웬 푸 쫑동지와 웰남사회주의공화국 주석 웬 쑤언 푹동지에게 답전을 보내시였다.

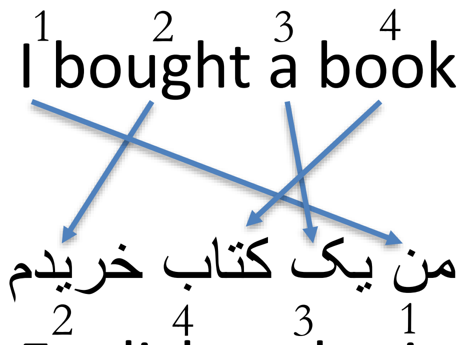
답전은 다음과 같다.

Dear Comrade Kim Jong-un, general secretary of the Workers' Party of Korea and Chairman of the State Affairs Commission of the Democratic People's Republic of Korea, sent a reply to Comrade Wen Phu Trong, general secretary of the Central Committee of the Vietnamese Communist Party, and Comrade Wen Xuan Phuc, President of the Socialist Republic of Vietnam on the 22nd.

  - Computer-aided translation: draft translation
  - Communication

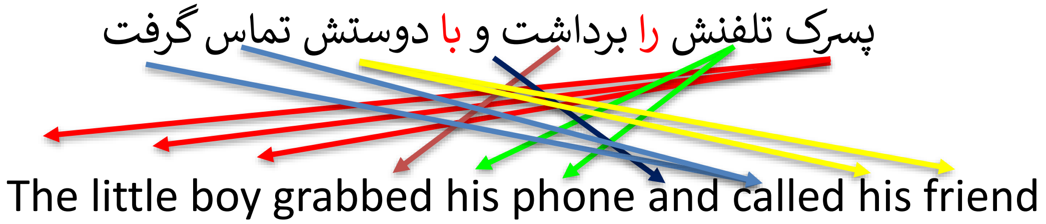
MT Challenges

- Lexical ambiguity:
- River Bank -> ساحل رودخانه
- Bank Account -> حساب بانکی
- Word order may be different



- I bought a book

- English order is SVO but Persian is SOV
- Syntactic Structure may not be preserved



پسرک تلفنش را برداشت و با دوستش تماس گرفت
The little boy grabbed his phone and called his friend

MT Challenges

- Cross-language Lexical ambiguity

Important

1. مهم 2. با اهمیت 3. عمده 4. خطیر 5. پراهمیت 6. ژرف 7. فوق العاده

- Syntactic ambiguity

I'm glad I’m healthy, and so is my child.

خوشحال هستم که سلامت هستم و همین طور خوشحالم که فرزندم سلامت است.

خوشحال هستم که سلامت هستم و همین طور فرزندم خوشحال است که من سلامت هستم

- Pronoun Resolution

او آمد

She/He came

Rule-Based Machine Translation

- The basis is on word-by-word translation
- No syntactic or semantic analysis is performed on the source language to resolve potential ambiguities
- We rely on a very large bilingual dictionary that allows for translating all words
- After translating the words, we use rule-based NLP to arrange the word order

Rule-Based Machine Translation

- Example: Machine translation and human being (Panov 1960s)

- Rules for translating much or many into Russian:

```
if preceding word is how return skol’ko
else if preceding word is as return stol’ko zhe
else if word is much
    if preceding word is very return nil
    else if following word is a noun return mnogo
else (word is many)
if preceding word is a preposition and following word is noun
    return mnogii
    else return mnogo
```

Challenges for Rule-Based Machine Translation

- Reordering words based on rules becomes very challenging for long sentences

یک کتاب آبی بزرگ پایه پنجم ابتدایی را تحویل گرفتم و به دانش آموز دادم

I got a big blue fifth grade elementary school and gave it to the student

- Lexical ambiguity may make the translation unmeaningful

I told him that he should study hard

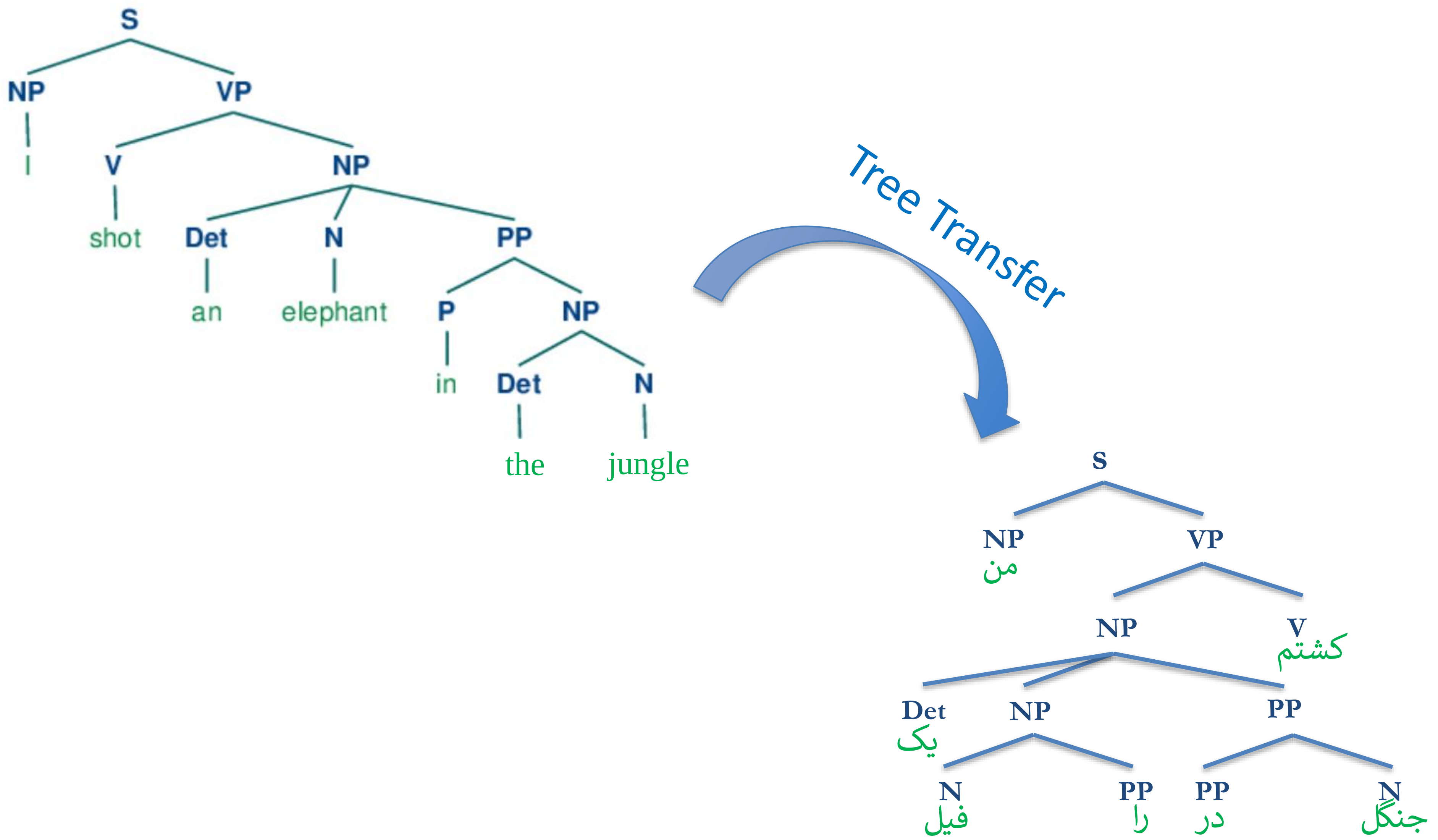
من به او گفتم که آن او باید مطالعه کند

Transfer-Based Machine Translation

- Improve Rule-based MT:
  - **Source analysis:** Analyze the source language: build a syntactic model for the source text
  - **Transfer:** Convert the source-language parse tree to a target-language parse tree.
  - **Generation:** Convert the target-language parse tree to an output sentence.

- Transfer stage is still rule-based but rules on syntactic structures are more generalizable, e.g., POS plus a set of rules based on the grammar of the source and the target language (SOV vs SVO)
  - Long sentence reordering is easier
  - SYSTRAN (Founded by Peter Toma in 1968) systems are based on this approach

Transfer-Based Machine Translation

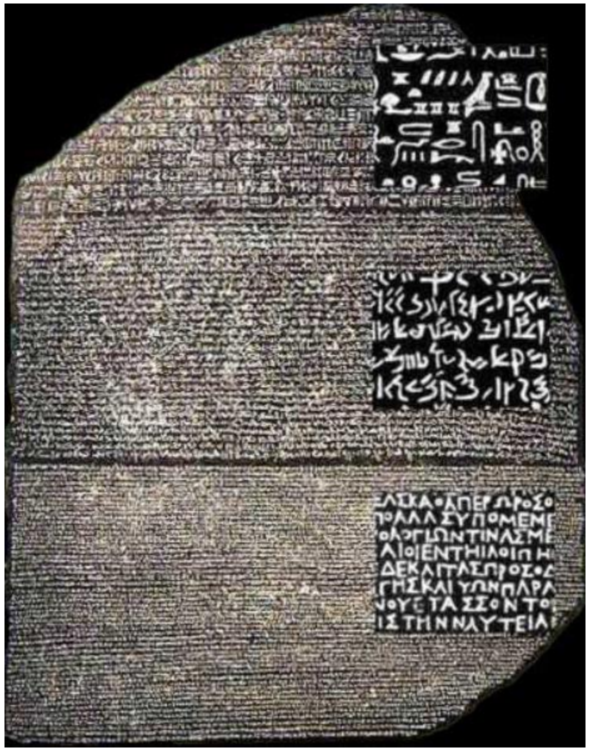


Statistical Machine Translation

- Idea: parallel corpora are available in several language pairs

- we can use a parallel corpus as a training set of translation examples and train a model to translate to relax the need for rules

- Old idea: Rosetta Stone



Statistical Machine Translation

- Parallel Texts:
  - Canadian Hansards: French-English (1.7 million sentences of 30 words or less in length), used by IBM
  - European Union
  - Translations of books
- MT Model:
  - Train a model that receives a sentence in the source language and returns the sentence in the target language
  - Use the model when the output is unknown
- IBM Models: 90s

The Noisy Channel Model for MT

- Goal: translate from French to English
- Generate a model p(e | f) which estimates conditional probability of any English sentence given the French sentence f.
- Use the training corpus to set the parameters.

- Noisy channel Model:

$$p(e \mid f) = \frac{p(e,f)}{p(f)} = \frac{\overset{\text{Language Model}}{p(e)}\,\overset{\text{Translation Model}}{p(f \mid e)}}{\sum_e p(e)p(f \mid e)}$$

Decoding Problem

$$\text{argmax}_e p(e \mid f) = \text{argmax}_e p(e)p(f \mid e)$$

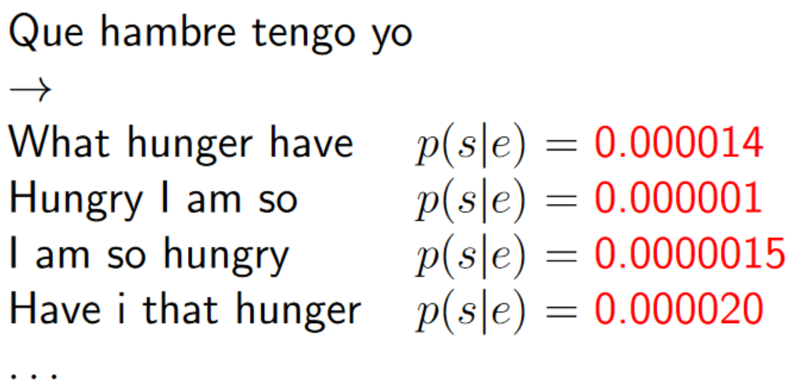
The Noisy Channel Model

- We can use a trigram as the language model

- It can be estimated using a larger English corpus

- The translation model is trained using the parallel corpus
- Example (from tutorial by Koehn and Knight)

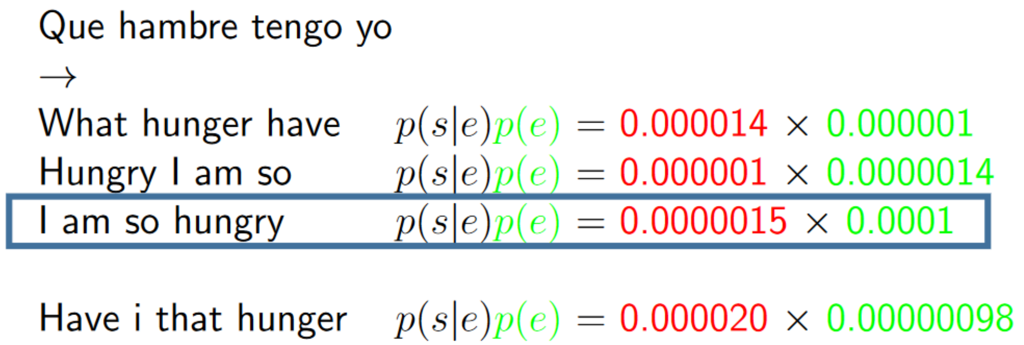
- Translation from Spanish to English



Que hambre tengo yo

$\rightarrow$

What hunger have	$p(s\|e) = 0.000014$
Hungry I am so	$p(s\|e) = 0.000001$
I am so hungry	$p(s\|e) = 0.0000015$
Have i that hunger	$p(s\|e) = 0.000020$
...	



Que hambre tengo yo

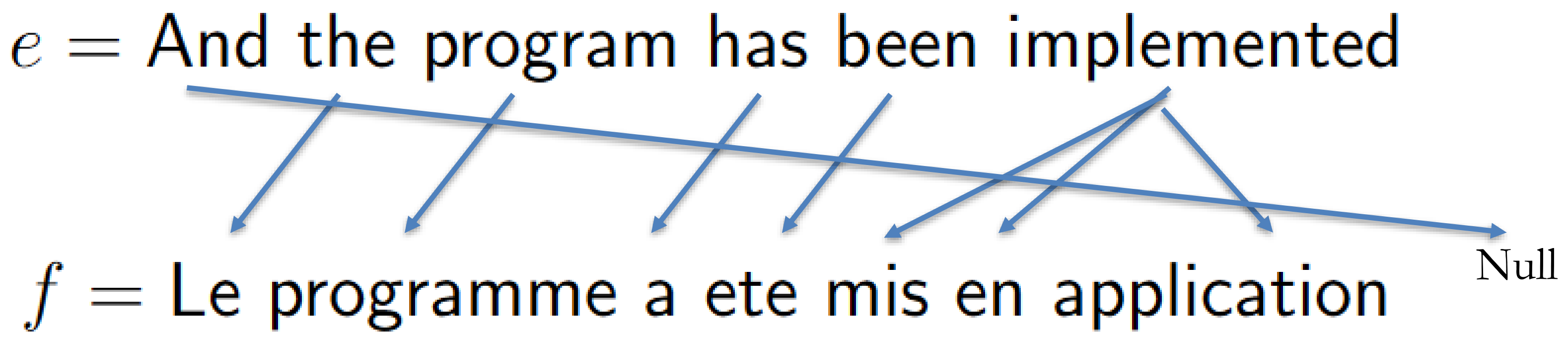
$\rightarrow$

What hunger have	$p(s\|e)p(e) = 0.000014 \times 0.000001$
Hungry I am so	$p(s\|e)p(e) = 0.000001 \times 0.0000014$
I am so hungry	$p(s\|e)p(e) = 0.0000015 \times 0.0001$
Have i that hunger	$p(s\|e)p(e) = 0.000020 \times 0.00000098$

Translation Model: IBM Model

- How do we model the translation model?

- In the parallel corpus, consider that for a pair, the English sentence has l words and the French sentence has m words
- An alignment map determines which English word each French word originated from
- An alignment a is $\{a_1, \dots a_m\}$, where $a_j \in \{0 \dots l\}$
- Hence there are $(l+1)^m$ possible alignments
- Ex: {2,3,4,5,6,6,6}


e = And the program has been implemented
f = Le programme a ete mis en application
Null

Translation Model

- Total probability over all possible alignments

$$p(f \mid e, m) = \sum_{a \in \mathcal{A}} \underbrace{\overset{\text{Alignment Distribution}}{p(a \mid e, m)} \overset{\text{Conditional Translation Model}}{p(f \mid a, e, m)}}_{p(f, a \mid e, m)}$$

- We will model the conditional probabilities:

$p(a \mid e, m)$ and $p(f \mid a, e, m)$

$$p(f, a \mid e, m) = p(a \mid e, m) p(f \mid a, e, m)$$

$$p(f \mid e, m) = \sum_{a \in \mathcal{A}} p(a \mid e, m) p(f \mid a, e, m)$$

- Having computed the conditional probabilities:

$$p(a \mid f, e, m) = \frac{p(f, a \mid e, m)}{\sum_{a \in \mathcal{A}} p(f, a \mid e, m)} \xleftarrow{} p(f \mid e, m)$$

Most Likely Alignment $$a^* = \arg\max_a p(a \mid f, e, m)$$

IBM Model 1

- Equally likely Alignment Probability

$$p(a \mid e, m) = \frac{1}{(l+1)^m}$$

- Conditional Translation Model

Model Parameters

$$p(f \mid a, e, m) = \prod_{j=1}^{m} t(f_j \mid e_{a_j})$$

- Ex: $l = 6,\ m = 7$ $a = \{2, 3, 4, 5, 6, 6, 6\}$

$e =$ And the program has been implemented

$f =$ Le programme a ete mis en application

$$p(f \mid a, e) = t(Le \mid the) \times t(programme \mid program) \times t(a \mid has) \times t(ete \mid been) \times \\ t(mis \mid implemented) \times t(en \mid implemented) \times t(application \mid implemented)$$

IBM Model 1

- An example of model parameters

English	French	Probability
position	position	0.756715
position	situation	0.0547918
position	mesure	0.0281663
position	vue	0.0169303
position	point	0.0124795
position	attitude	0.0108907

IBM Model 1: Generative Process

- Pick an alignment randomly: $\frac{1}{(l+1)^m}$

- Pick the corresponding French words

$$p(f \mid a, e, m) = \prod_{j=1}^{m} t(f_j \mid e_{a_j})$$

- Compute the conditional translation probability

$$p(f, a \mid e, m) = p(a \mid e, m) \times p(f \mid a, e, m) = \frac{1}{(l+1)^m} \prod_{j=1}^{m} t(f_j \mid e_{a_j})$$

IBM Model 2

- Non-uniform alignments: distortion parameters

Model 1

$$p(a \mid e, m) = \frac{1}{(l+1)^m}$$

Model 2

$$p(a \mid e, m) = \prod_{j=1}^{m} \mathbf{q}(a_j = i \mid j, l, m)$$

j's French word is generates from i's English word given the lengths

- Conditional Translation Model

Model 2

$$p(f, a \mid e, m) = \prod_{j=1}^{m} \mathbf{q}(a_j \mid j, l, m)\mathbf{t}(f_j \mid e_{a_j})$$

Model 1

$$p(f \mid a, e, m) = \prod_{j=1}^{m} t(f_j \mid e_{a_j})$$

IBM Model 2

- Example

$$
\begin{aligned}
l &= 6 \\
m &= 7 \\
e &= \text{And the program has been implemented} \\
f &= \text{Le programme a ete mis en application} \\
a &= \{2, 3, 4, 5, 6, 6, 6\}
\end{aligned}
$$

$$
\begin{aligned}
p(a \mid e, 7) = {} & \mathbf{q}(2 \mid 1, 6, 7) \times \mathbf{q}(3 \mid 2, 6, 7) \times \mathbf{q}(4 \mid 3, 6, 7) \times \\
& \mathbf{q}(5 \mid 4, 6, 7) \times \mathbf{q}(6 \mid 5, 6, 7) \times \mathbf{q}(6 \mid 6, 6, 7) \times \mathbf{q}(6 \mid 7, 6, 7)
\end{aligned}
$$

$$
\begin{aligned}
p(f \mid a, e, 7) = {} & t(Le \mid the) \times t(programme \mid program) \times t(a \mid has) \times t(ete \mid been) \times \\
& t(mis \mid implemented) \times t(en \mid implemented) \times t(application \mid implemented)
\end{aligned}
$$

IBM Model 2: Generative Process

- Pick an alignment randomly: $\prod_{j=1}^{m} \mathbf{q}(a_j \mid j, l, m)$

- Pick the corresponding French words

$$p(f \mid a, e, m) = \prod_{j=1}^{m} \mathbf{t}(f_j \mid e_{a_j})$$

- Compute the conditional translation probability

$$p(f, a \mid e, m) = p(a \mid e, m) p(f \mid a, e, m) = \prod_{j=1}^{m} \mathbf{q}(a_j \mid j, l, m) \mathbf{t}(f_j \mid e_{a_j})$$

IBM Model Parameter Estimation

- Input: sentence pairs $(e^{(k)}, f^{(k)})$
- Output: parameters $t(f|e)$ and $q(i|j,l,m)$
- Primary Challenge: alignments are not known

- Data annotation is expensive

$$\begin{aligned} e^{(100)} &= \text{And the program has been implemented} \\ f^{(100)} &= \text{Le programme a ete mis en application} \end{aligned}$$

- Expectation Maximization (EM) algorithm

IBM Model Parameter Estimation

- Assume the alignments are accessible

$$\begin{aligned} e^{(100)} &= \text{And the program has been implemented} \\ f^{(100)} &= \text{Le programme a ete mis en application} \\ a^{(100)} &= \langle 2, 3, 4, 5, 6, 6, 6 \rangle \end{aligned}$$

- We will have triplets $(e^{(k)}, f^{(k)}, a^{(k)})$
- ML estimates for parameters boils down to counting, ex, t(position|position)

$$t_{ML}(f|e) = \frac{\text{Count}(e,\ f)}{\text{Count}(e)} \qquad q_{ML}(j|i,l,m) = \frac{\text{Count}(j,i,l,m)}{\text{Count}(i,l,m)}$$

IBM Model Parameter Estimation

Ex:
e= the position
f=La position
a = {1,2}

Input: A training corpus $(f^{(k)}, e^{(k)}, a^{(k)})$ for $k = 1 \dots n$, where $f^{(k)} = f_1^{(k)} \dots f_{m_k}^{(k)}$, $e^{(k)} = e_1^{(k)} \dots e_{l_k}^{(k)}$, $a^{(k)} = a_1^{(k)} \dots a_{m_k}^{(k)}$.

Algorithm:

- Set all counts $c(\dots) = 0$
- For $k = 1 \dots n$
  - For $i = 1 \dots m_k$, For $j = 0 \dots l_k$,

$$\begin{aligned} c(e_j^{(k)}, f_i^{(k)}) &\leftarrow c(e_j^{(k)}, f_i^{(k)}) + \delta(k,i,j) \\ c(e_j^{(k)}) &\leftarrow c(e_j^{(k)}) + \delta(k,i,j) \\ c(j|i,l,m) &\leftarrow c(j|i,l,m) + \delta(k,i,j) \\ c(i,l,m) &\leftarrow c(i,l,m) + \delta(k,i,j) \end{aligned}$$

English Position

French Position

where $\delta(k,i,j) = 1$ if $\boxed{a_i^{(k)} = j, 0}$ otherwise.

Pair Index

Output: $t_{ML}(f|e) = \frac{c(e,f)}{c(e)}$, $q_{ML}(j|i,l,m) = \frac{c(j|i,l,m)}{c(i,l,m)}$

Expectation Maximization

- Dempster et al., 1977: An algorithm for computing maximum likelihood from incomplete data:
- incomplete data means that data annotation is not complete to allow for estimating the model paramters
- if we had complete data, would could estimate model
- if we had model, we could fill in the gaps in the data
- EM in a nutshell:

1. initialize model parameters, e.g., random
2. assign probabilities to the missing data
3. estimate model parameters from completed data
4. iterate steps 2–3 until convergence

EM Algorithm for MT

- We don't have the alignments:

1. The algorithm is **iterative**: we start with some arbitrary random choice for the q and t parameters. At each iteration we compute the "counts" based on the data together with our current parameter estimates. We then re-estimate the parameters with these counts, and iterate
2. $\delta(k,i,j)$ is computed as follows

$$\delta(k,i,j) = \frac{q(j|i,l_k,m_k)t(f_i^{(k)}|e_j^{(k)})}{\sum_{j=0}^{l_k} q(j|i,l_k,m_k)t(f_i^{(k)}|e_j^{(k)})}$$

EM Algorithm for MT

- S ~ 10-20
- Delta parameters:

$$\delta(k,i,j) = P(a_j^{(k)} = i | e^{(k)}, f^{(k)})$$

- EM would converge to local ML optimums

For $s = 1 \ldots S$

- ▸ Set all counts $c(\ldots) = 0$
- ▸ For $k = 1 \ldots n$
    - ▸ For $i = 1 \ldots m_k$, For $j = 0 \ldots l_k$

M-Step

$$\begin{aligned} c(e_j^{(k)}, f_i^{(k)}) &\leftarrow c(e_j^{(k)}, f_i^{(k)}) + \delta(k,i,j) \\ c(e_j^{(k)}) &\leftarrow c(e_j^{(k)}) + \delta(k,i,j) \\ c(j|i,l,m) &\leftarrow c(j|i,l,m) + \delta(k,i,j) \\ c(i,l,m) &\leftarrow c(i,l,m) + \delta(k,i,j) \end{aligned}$$

where

E-Step

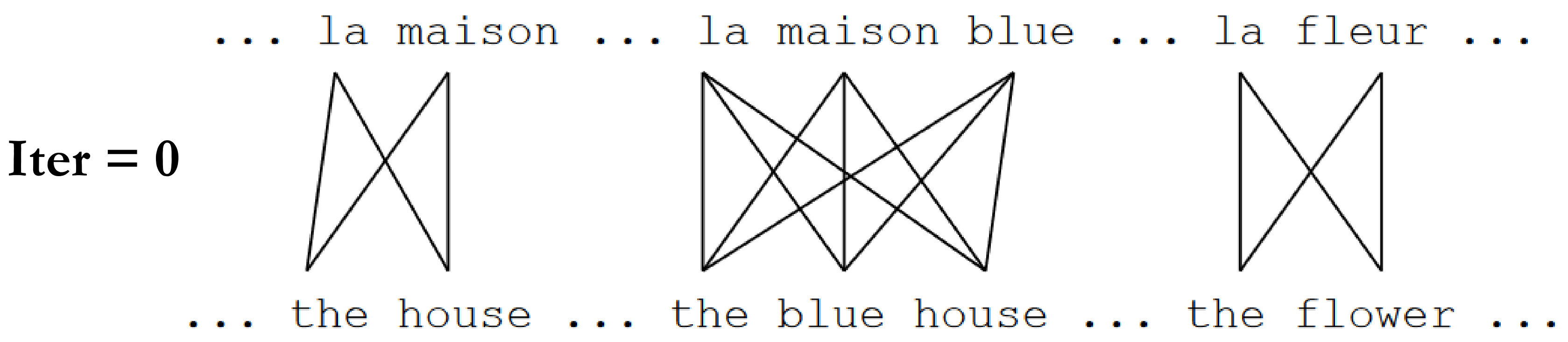
$$\delta(k,i,j) = \frac{q(j|i,l_k,m_k)t(f_i^{(k)}|e_j^{(k)})}{\sum_{j=0}^{l_k} q(j|i,l_k,m_k)t(f_i^{(k)}|e_j^{(k)})}$$

- ▸ Recalculate the parameters:

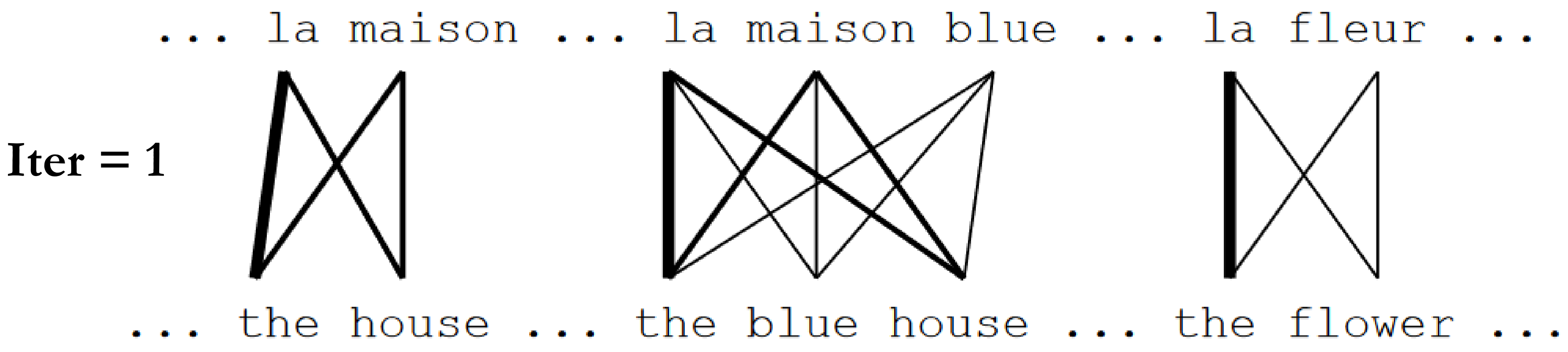
$$t(f|e) = \frac{c(e,f)}{c(e)} \qquad q(j|i,l,m) = \frac{c(j|i,l,m)}{c(i,l,m)}$$

EM Algorithm for MT

- Initialization: set all assignments equally likely
- Model learns ‘La’ is often aligned with ‘the’



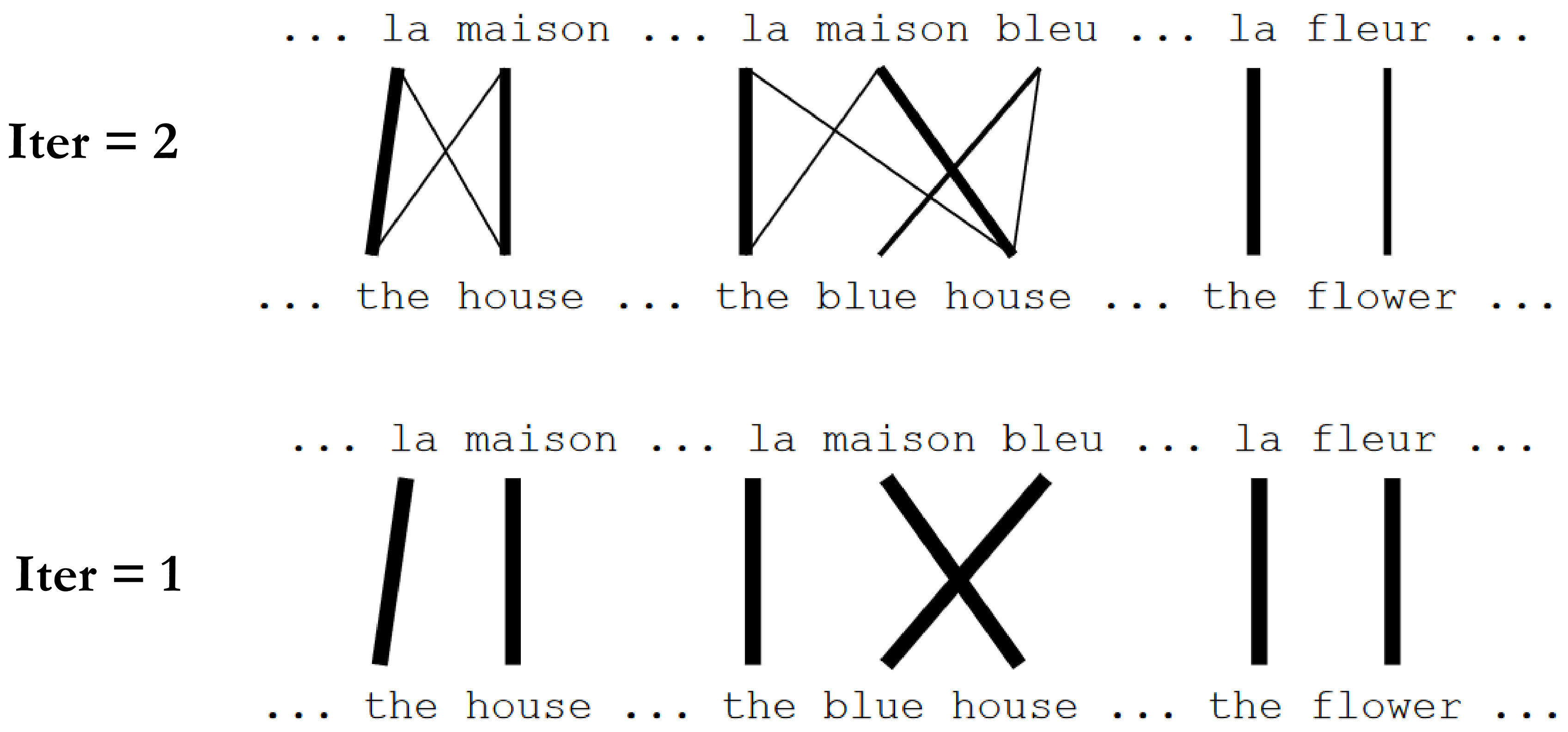
Iter = 0
... la maison ... la maison blue ... la fleur ...
... the house ... the blue house ... the flower ...



Iter = 1
... la maison ... la maison blue ... la fleur ...
... the house ... the blue house ... the flower ...

EM Algorithm for MT

- After one more iteration `fleur' is aligned with 'flower'
- Convergence: after One more iteration



Iter = 2
... la maison ... la maison bleu ... la fleur ...
... the house ... the blue house ... the flower ...
Iter = 1
... la maison ... la maison bleu ... la fleur ...
... the house ... the blue house ... the flower ...

Model Evaluation

- Perplexity: derived from probability of the training data according to the model

$$\log_2 PP = -\sum_s \log_2 p(\mathbf{e}_s|\mathbf{f}_s)$$

- Ex:

	initial	1st it.	2nd it.	3rd it.	...	final
p(the haus|das haus)	0.0625	0.1875	0.1905	0.1913	...	0.1875
p(the book|das buch)	0.0625	0.1406	0.1790	0.2075	...	0.25
p(a book|ein buch)	0.0625	0.1875	0.1907	0.1913	...	0.1875
perplexity	4095	202.3	153.6	131.6	...	113.8