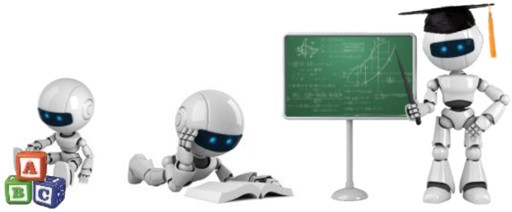
USC Viterbi
School of Engineering
Department of Computer Science

CSCI 544
Applied Natural Language Processing

Mohammad Rostami
USC Computer Science Department

Slides adopted from slides by Jurfsky, Martin, and Collins



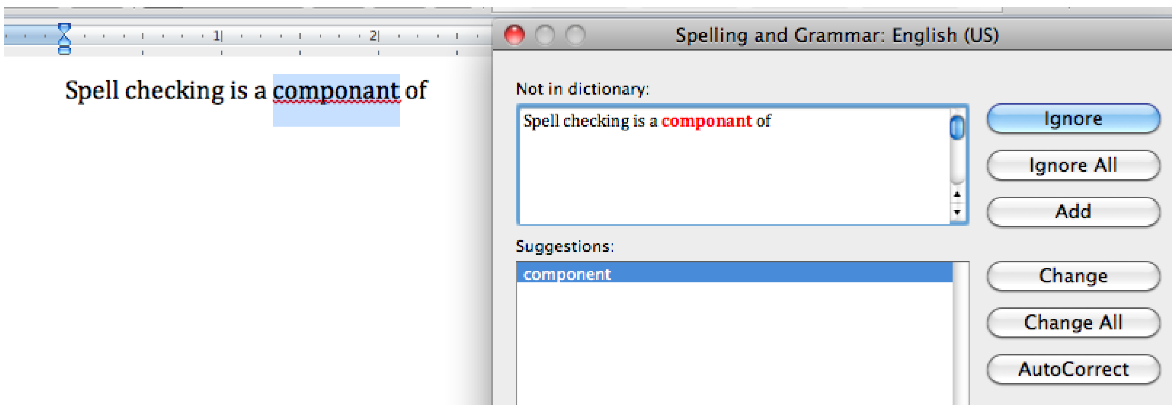
Logistical Notes

- Next Quiz
- HW4:
  - Three weeks
  - Self-learning may be necessary
- Project Advisor Meetings
- Status Report
- Group Activity Grades
- Midterm Stat: Average is 451.5 and STD ~30

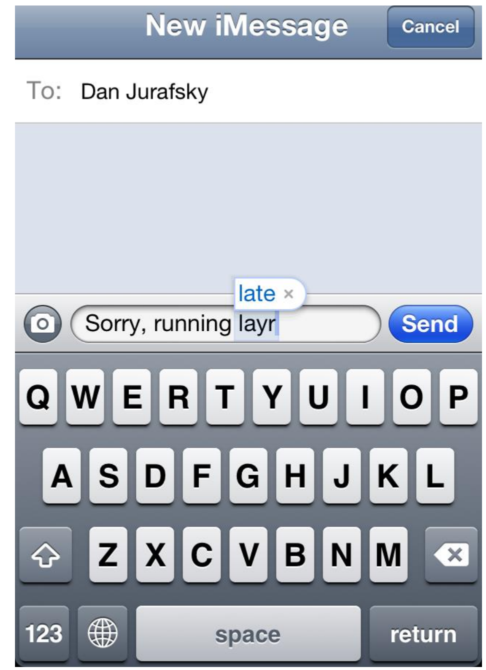
Spelling Correction

- Applications

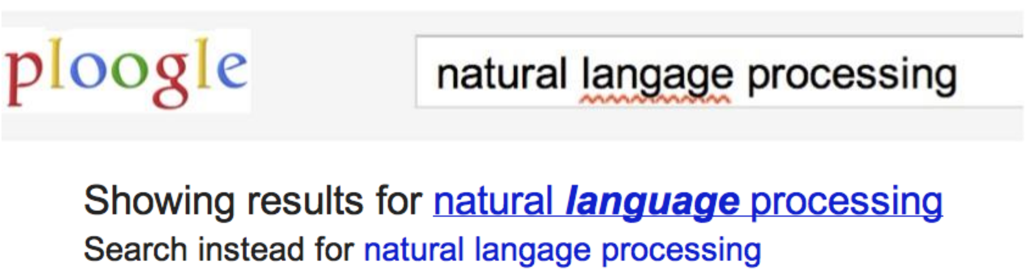
Word processing



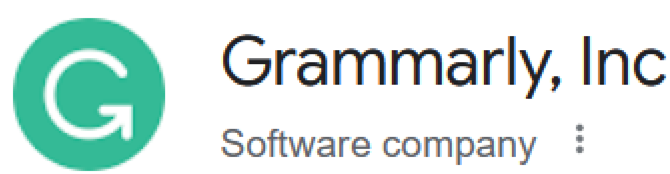
Phones



Web search



Grammarly



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

- **2**%: Words uncorrected on organizer Soukoreff &MacKenzie 2003

- **1-2**%: Retyping: Kane and Wobbrock 2007, Gruden et al. 1983

Spelling Correction

- Spelling Correction 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*, compute the probability of being meaningful in the context, and if an error is detected, generate a candidate set:
  - Find candidate words with similar ***pronunciations***
  - Find candidate words with similar ***spelling***
  - Include *w* in candidate set
- Choose best candidate
  - Noisy Channel

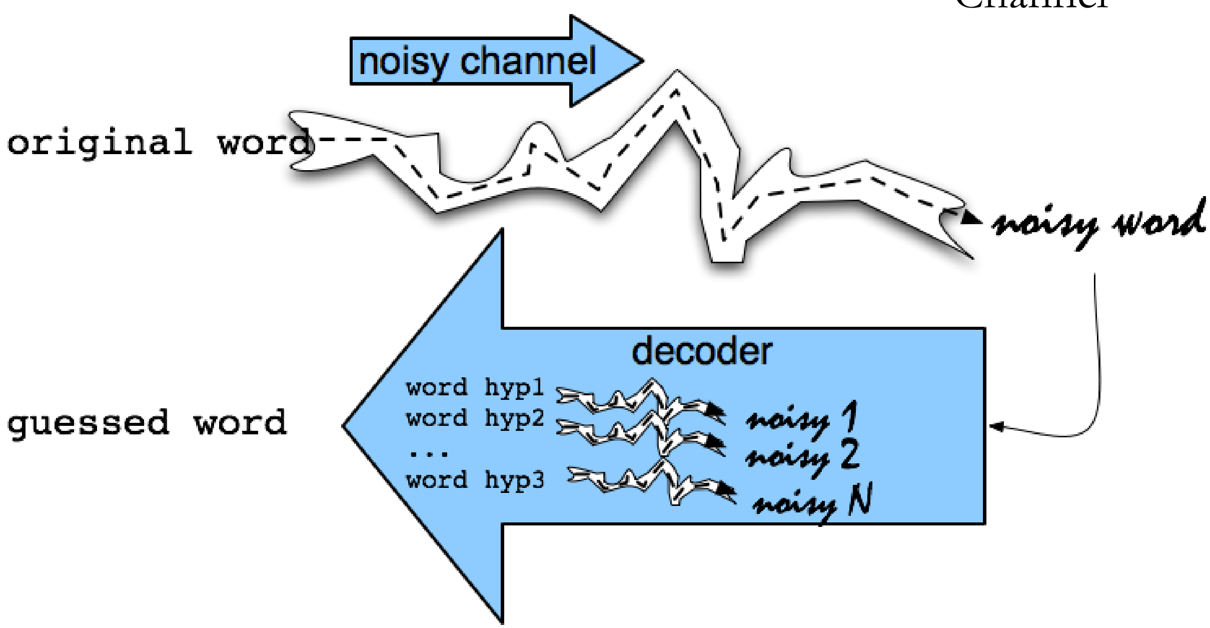
Spelling Correction: Noisy Channel Model

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

$$\hat{w} = \underset{w \in V}{\operatorname{argmax}} P(w \mid x) \qquad = \underset{w \in V}{\operatorname{argmax}} \frac{P(x \mid w)P(w)}{P(x)} \qquad = \underset{w \in V}{\operatorname{argmax}} P(x \mid w)P(w)$$

Channel

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
  - Deletion
  - Substitution
  - Transposition of two adjacent letters

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`
- We can use brute-search to find all candidates by considering all one-letter transforms
- Language Model: AP corpus (N = 44 million words)

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

Spelling Error Patterns

- Spelling errors are character-level errors

- Vowels: ie vs ei -> receive vs recieve
- Keys close to each other on keyboard: a and s -> About vs Sbout
- Spaces: New York vs NewYork
- Shuffles: Machine vs Mahcine

- We can model spelling errors across all words in the corpus

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

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

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

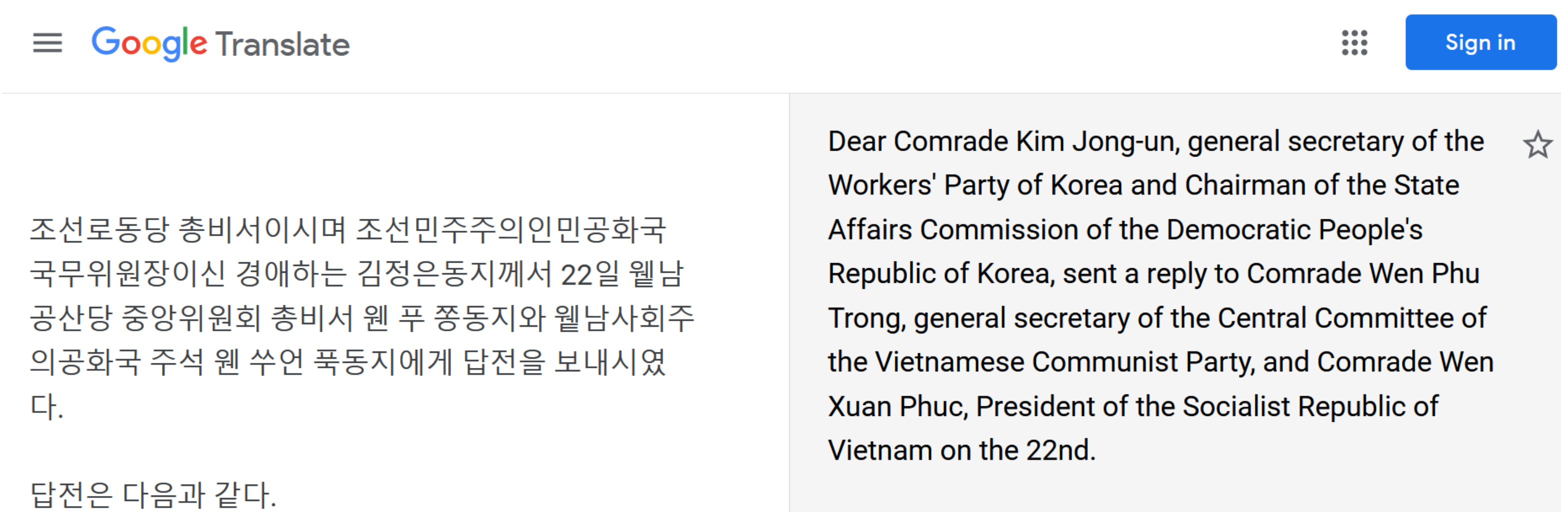
- P("versatile actress whose") = .000021*.0010 = 210 $\times 10^{-10}$
- P("versatile across whose") = .000021*.000006 = 1 $\times 10^{-10}$

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



  - Computer-aided translation: draft translation
  - Communication

MT Challenges

- Lexical ambiguity:
  - River Bank -> ساحل رودخانه
  - Bank Account -> حساب بانکی
- Word order may be different
  - 1 2 3 4
    I bought a book

    من یک کتاب خریدم
    2 4 3 1
  - 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

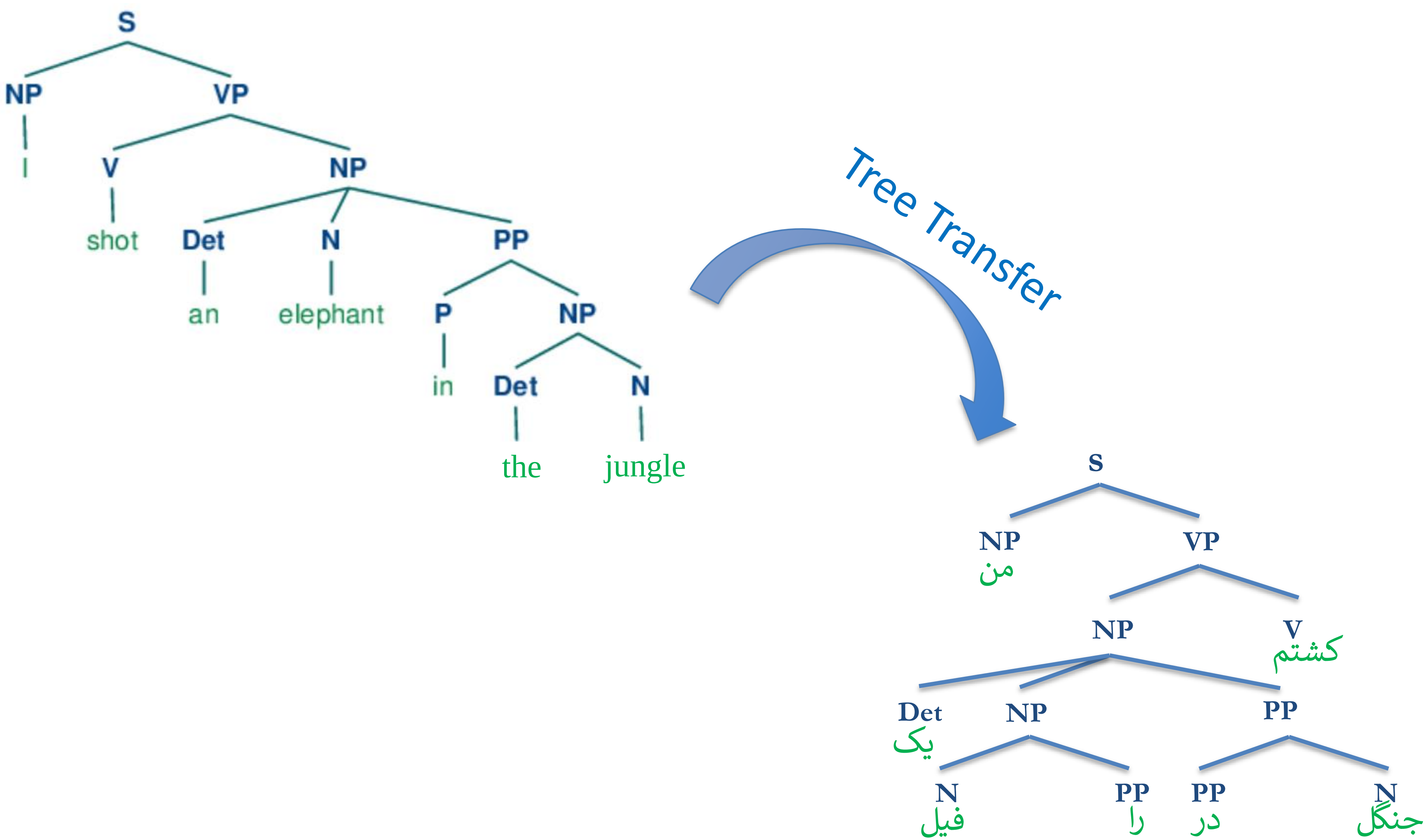
من به او گفتم که آن او باید مطالعه کند

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: Noisy Channel 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:

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

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	$p(s\|e) = 0.000020$

...

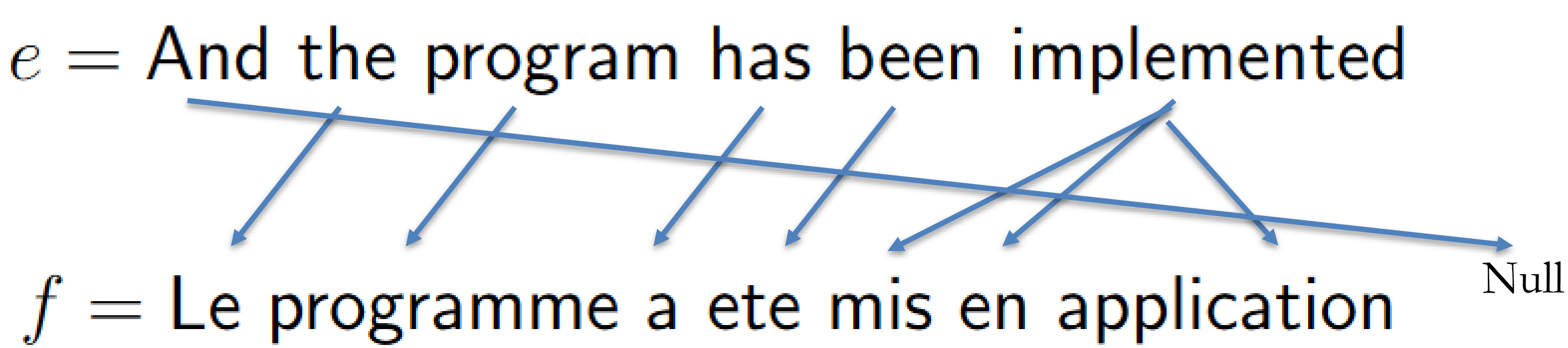
Que hambre tengo yo

$\rightarrow$

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 $\{a_1, \ldots a_m\}$, where $a_j \in \{0 \ldots l\}$
- Hence there are $(l+1)^m$ possible alignments
- Ex: {2,3,4,5,6,6,6,0}



Translation Model

- Total probability over all possible alignments

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

- We will model the conditional probabilities:

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

- We need to compute these two probabilities using the parallel corpus

IBM Model 1

- Equally likely Alignment Probability

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

- Conditional Translation Model

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

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

$$e = \text{And the program has been implemented}$$
$$f = \text{Le programme a ete mis en application}$$

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

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

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