Machine Translation

Peerachet Porkaew, NECTEC

Outline

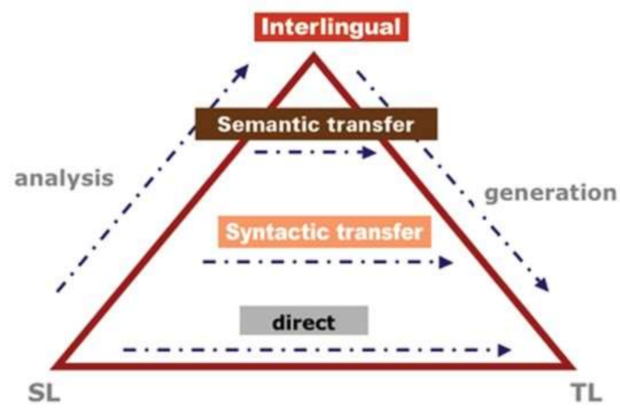
- Rule-based Machine Translation
- Statistical Machine Translation (SMT)

Rule-based Machine Translation

Rule-based Machine Translation (RBMT)

- Linguistic Knowledge
- Three sub-module :
    - Analysis
    - Transfer
    - Generation
- Template-based Translation

Rule-based MT

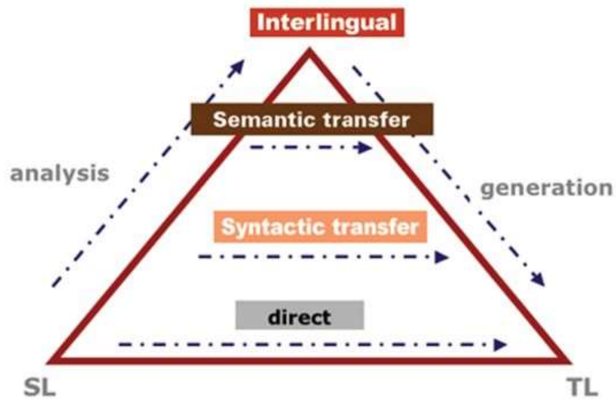


The Vauquois Triangle

Rule-based Machine Translation (RBMT)

- Limitations :
  - Not Automatic
  - Time consuming
  - Conflicts
  - Less Clarity

Rule-based MT



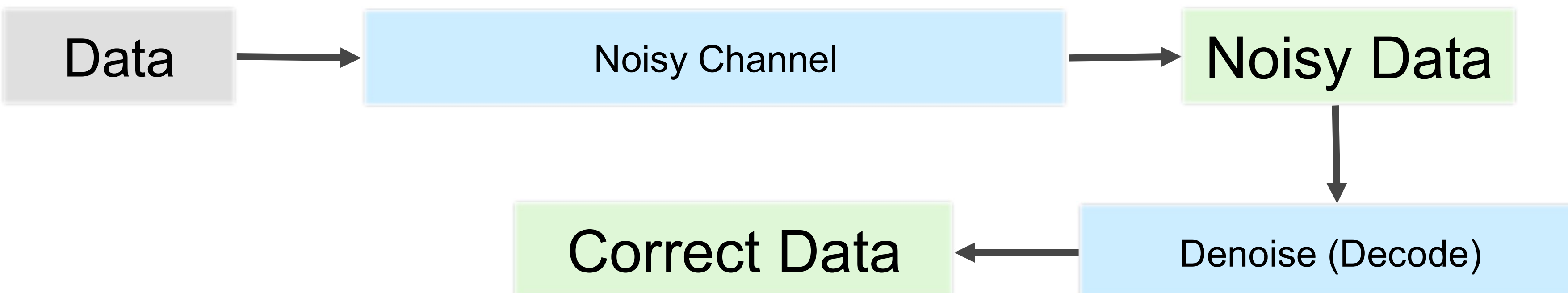
The Vauquois Triangle

Simple RBMT Demo

Statistical Machine Translation

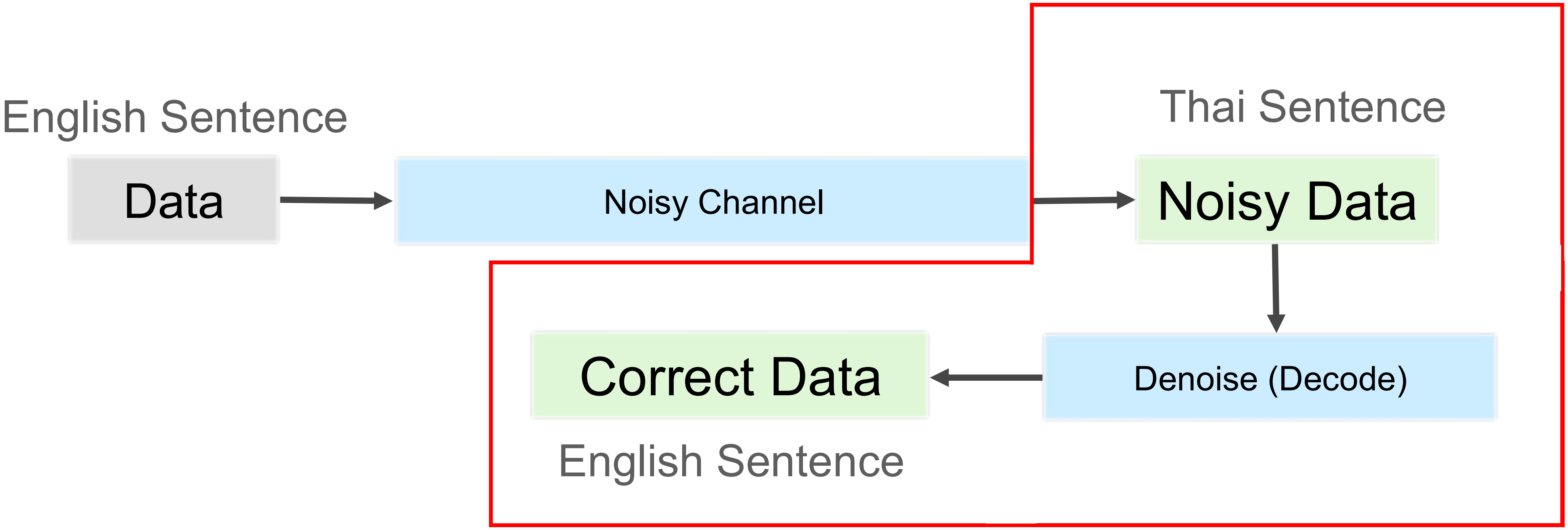
Statistical Machine Translation

- Noisy Channel Model



Statistical Machine Translation

- Noisy Channel Model



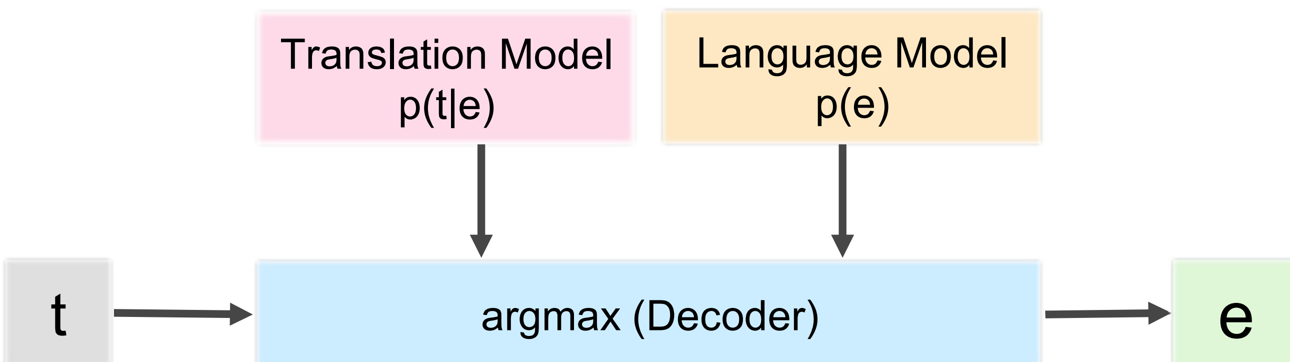
Statistical Machine Translation

- Source sentence ***t***, e.g. Thai
- Target sentence ***e***, e.g. English
- Probabilistic formulation using Bayes rule $P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$

$$\hat{e} = \mathrm{argmax}_e\, \mathrm{p}(e|t)$$
$$\hat{e} = \mathrm{argmax}_e\, \mathrm{p}(t|e)p(e)$$

Statistical Machine Translation (Cont.)

$$\hat{e} = \mathrm{argmax}_e\, \mathrm{p}(t|e)p(e)$$



Statistical Decoder (Simplified version)

เมื่อวาน	นี้	ฉัน	ไป	ทะเล	กับ	เพื่อน
Yesterday	this	I	go	the sea	with	friends
Yesterday		I	went to	sea	with	friend
Previous day		me	get to	ocean	with my friend	

เมื่อวาน	นี้	ฉัน	ไป	ทะเล	กับ	เพื่อน
Yesterday	this	I	go	the sea	with	friends
Yesterday		I	went to	sea	with	friend
Previous day		me	get to	ocean	with my friend	

เมื่อวาน	นี้	ฉัน	ไป	ทะเล	กับ	เพื่อน
Yesterday	this	I	go	the sea	with	friends
Yesterday		I	went to	sea	with	friend
Previous day		me	get to	ocean	with my friend	

Translation Model *p*(t|e)

- Word-based Translation Model

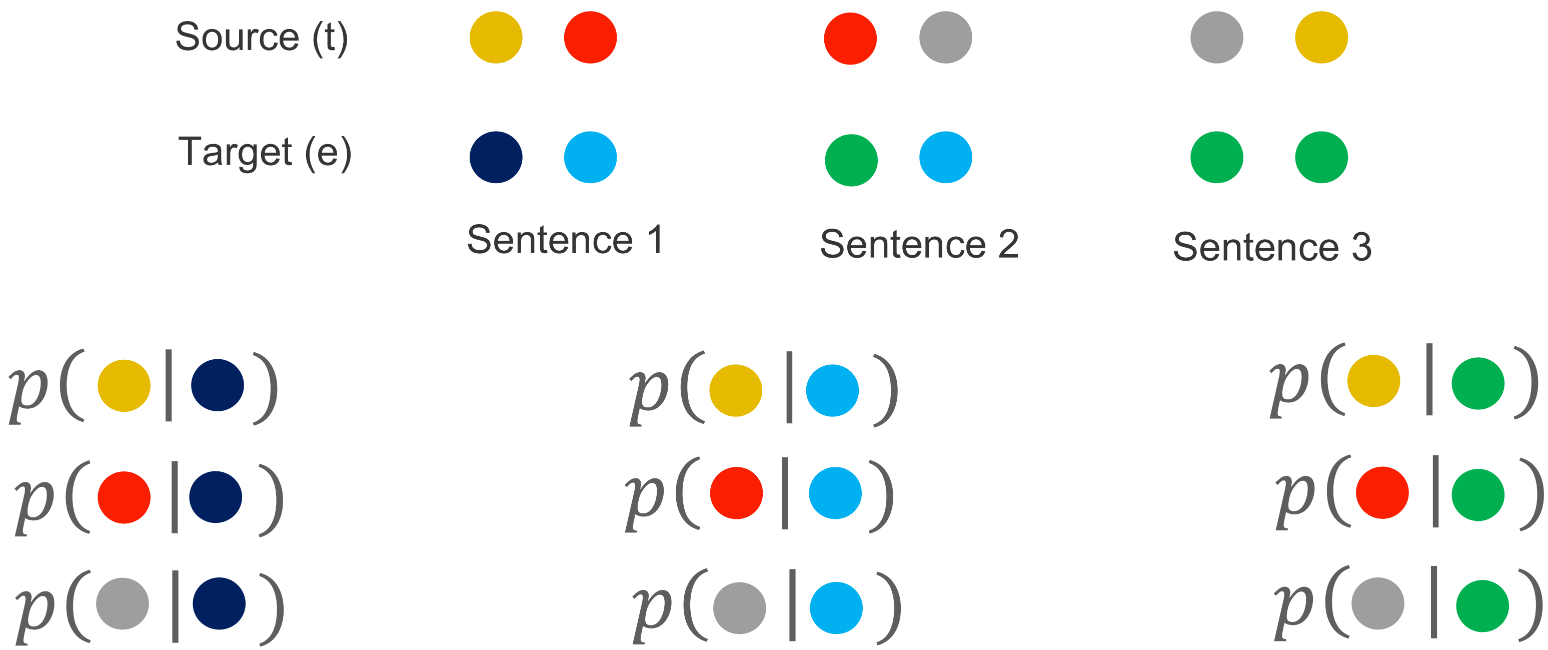
$$p(t|e) = p(\text{ไป}|go) = 0.5$$

- Phrase-based Translation Model

$$\mathrm{p}(t_1 t_2, \dots, t_n | e_1, e_2, \dots, e_m) = p(\text{เมื่อวาน นี้}|\text{ previous day}) = 0.1$$

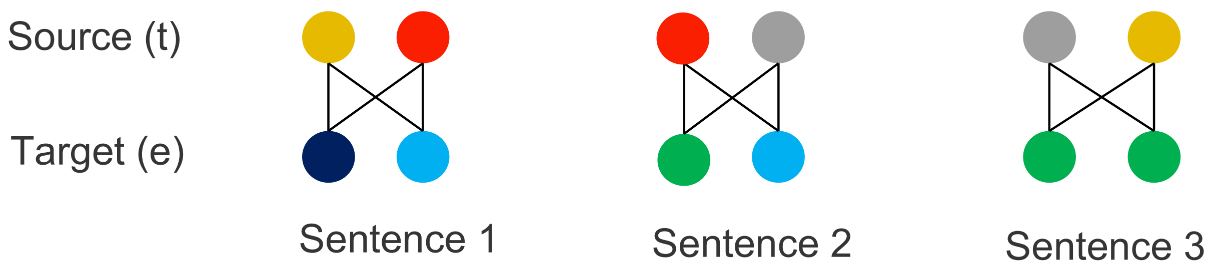
How we get the *p*(t|e) ?

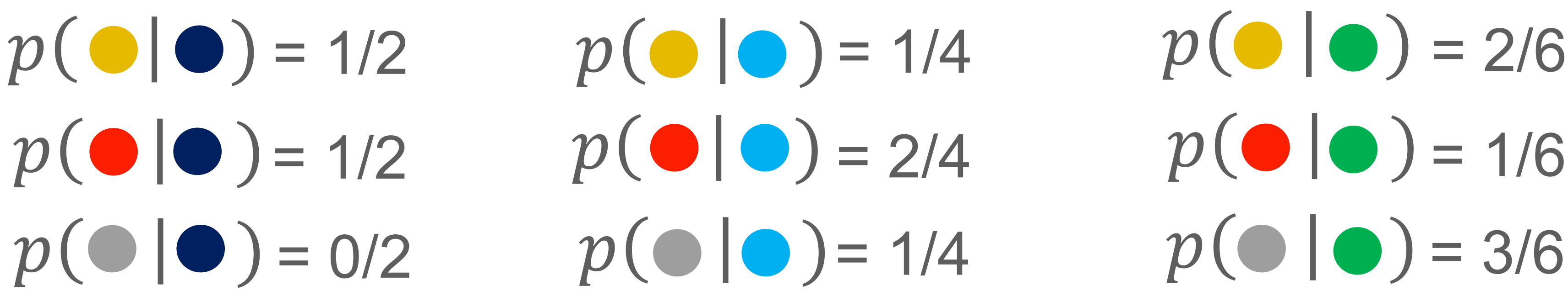
- We do not have alignments. No problem we assume alignment are uniformly paired. ***p(t|e) = c*** (constant)



How we get the *p*(t|e) ?

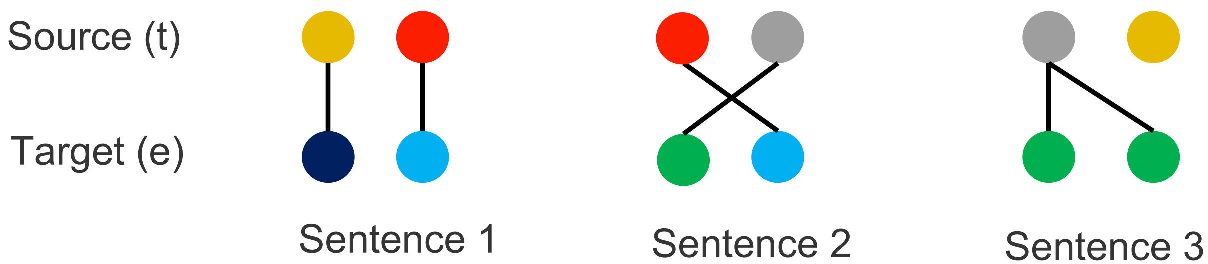
- But, we do not have alignment. No problem we assume alignment are uniformly paired. ***p(t|e) = c*** (constant)





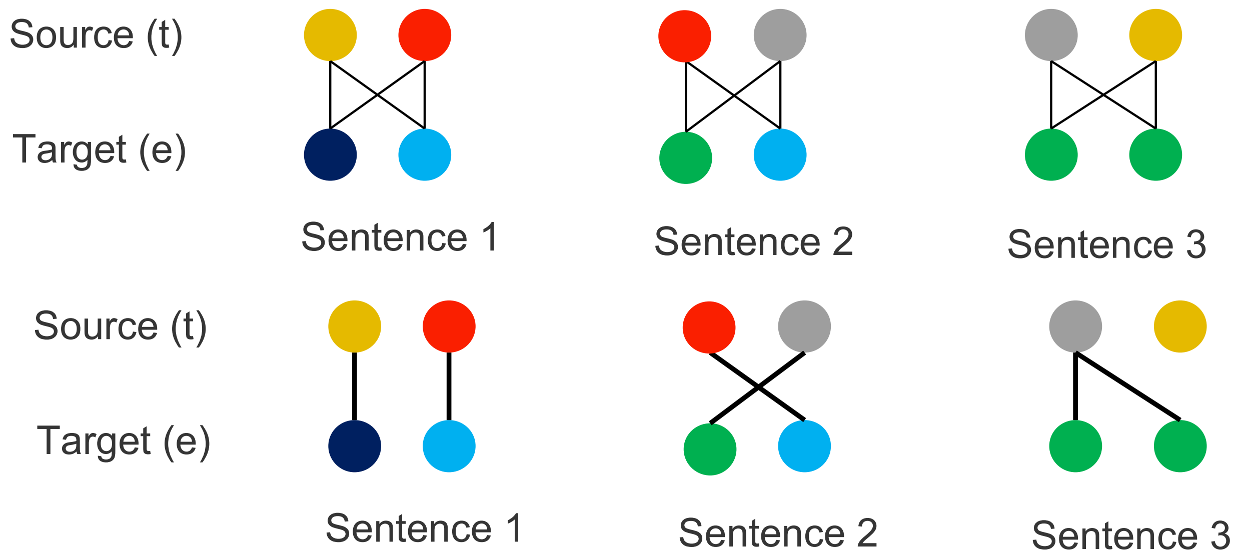
How we get the *p*(t|e) ?

- Now, we can get the better alignment from previous knowledge.



Then, we can calculate *p(t|e)* again using these alignment information.

How we get the *p*(t|e) ?



Expectation Maximization (EM)

Translation Model

- **Word-based Translation Model**

$$p(t|e) = p(\text{ไป}|go) = 0.5$$

- Phrase-based Translation Model

p($t_1 t_2, \ldots, t_n | e_1, e_2, \ldots, e_m$)= p(เมื่อวาน นี้| previous day) = 0.1

Word-based Translation Model

- **Word-based Translation Model**

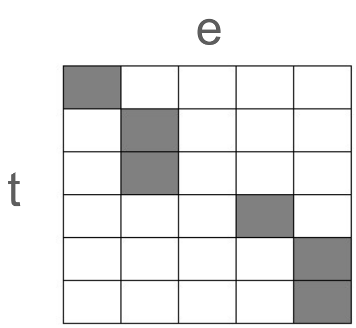
$$p(t|e) = p(\text{ไป}|go) = 0.5$$

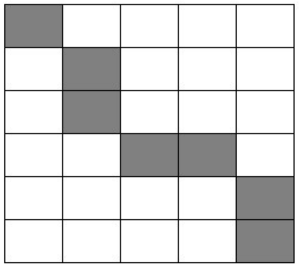
- **Phrase-based Translation Model**

$$\mathrm{p}(t_1 t_2, \dots, t_n | e_1, e_2, \dots, e_m) = p(\text{เมื่อวาน นี้}| \text{ previous day}) = 0.1$$

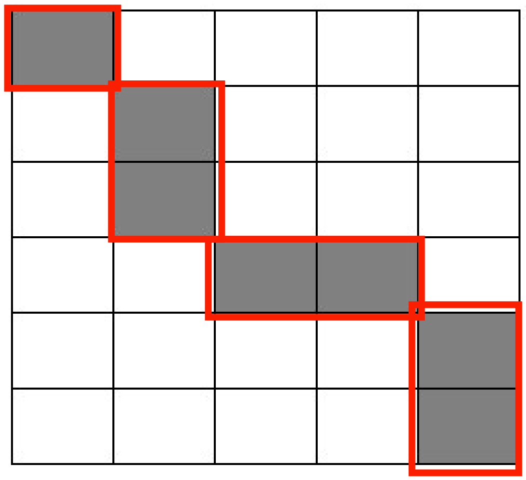
How we get the P(t|e) for phrases ?

- Phrase Extraction Algorithm
  - Expanding single word alignment pairs to multiple-word alignment.



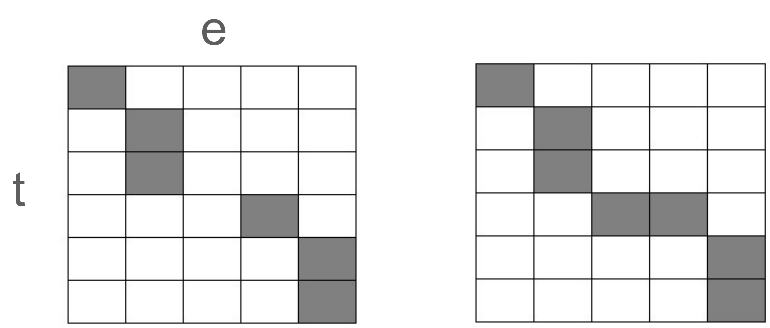


Fill missing alignment pairs

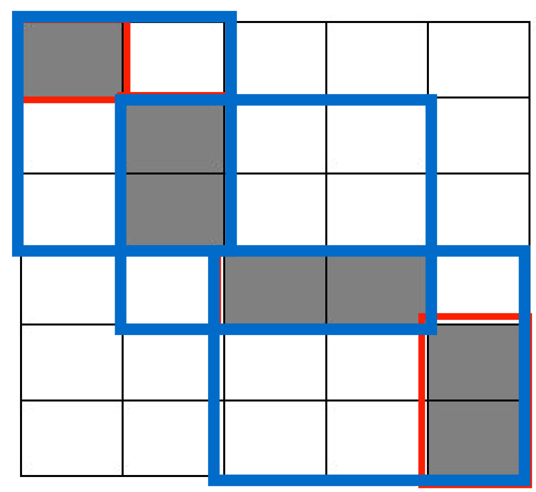


How we get the P(t|e) for phrases ?

- Phrase Extraction Algorithm
  - Expanding single word alignment pairs to multiple-word alignment.



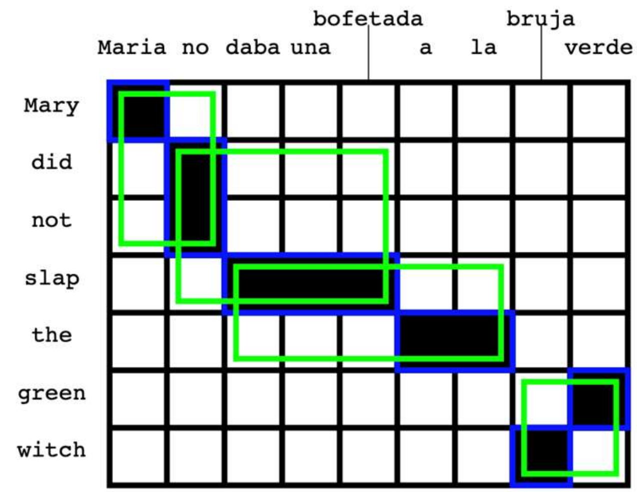
Fill missing alignment pairs



How we get the P(t|e) for phrases ?

- Phrase Pairs

$$p(t|e) = \frac{count(t,e)}{count(e)}$$



(Maria, Mary), (no, did not), (slap, daba una bofetada), (a la, the), (bruja, witch), (verde, green), (Maria no, Mary did not), (no daba una bofetada, did not slap), (daba una bofetada a la, slap the), (bruja verde, green witch)

Image from : www.statmt.org

Statistical Decoder (with Translation model)

เมื่อวาน	นี้	ฉัน	ไป	ทะเล	กับ	เพื่อน
Yesterday (0.5)	this (1.0)	I (0.8)	go (0.6)	the sea (0.7)	with (0.8)	friends (0.6)
Yesterday (0.7)		I (0.8)	went to (0.3)	sea (0.2)	with (0.8)	friend (0.4)
Previous day (0.1)		me (0.2)	get to (0.01)	ocean (0.05)	with my friend (0.1)	

Language Model $p_{LM}(e)$

- Language Model of the target language.
- Calculate the ***"fluency"*** of sentences

$$p_{LM}(\text{I went to the sea}) > p_{LM}(\text{I went to ocean})$$

How can we estimate p_{LM} ?

- **N-gram** language model

1-gram : p_{LM}(I went to the sea) =

$$p(I) \times p(went) \times p(to) \times p(the) \times p(sea)$$

2-gram : p_{LM}(I went to the sea) =

$$p(I \mid \text{<bos>}) \times p(went \mid I) \times p(to \mid went) \times p(the \mid to) \times p(sea \mid the)$$

How can we estimate P_{LM} ?

- **Maximum Likelihood Estimation (MLE)**

- p(I) = count("I") / N

- p(went|I) = count("I went") / count("I")

Smoothing

- if p(x|y) = 0 ? ➔ $P_{LM} = 0$

We can use smoothing technique to overcome this situation.

For example, **Add-one smoothing (Laplace Smoothing)**

$$P^*_{\text{Laplace}}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{\sum_w (C(w_{n-1}w)+1)} = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

Smoothing

- **Back-off**

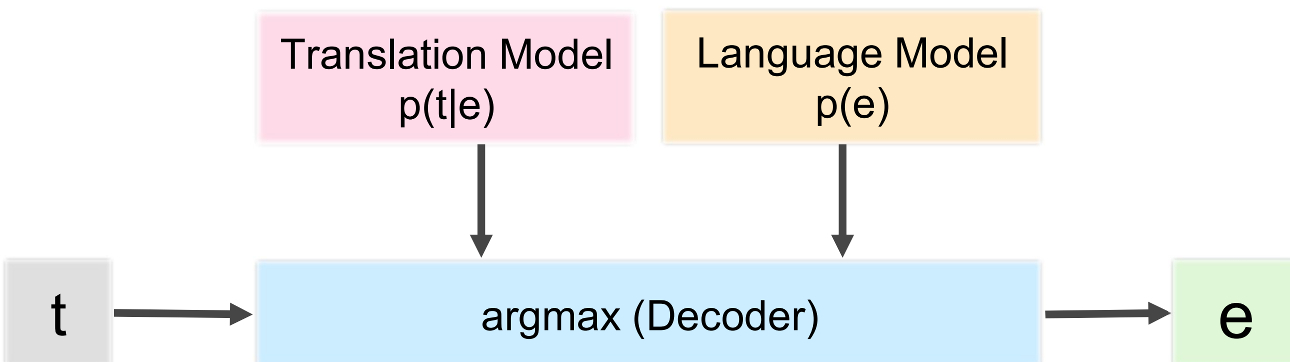
if $p(w_i|w_{i-1}) = 0$ then $\hat{p}(w_i|w_{i-1}) = p(w_i)$

- **Interpolation**

$$\hat{p}(w_i|w_{i-1}) = \lambda_1 p(w_i) + \lambda_2 p(w_i|w_{i-1})$$
$$\lambda_1 + \lambda_2 = 1$$

Statistical Decoder

$$\hat{e} = \text{argmax}_e \, \mathrm{p}(t|e)p(e)$$



Decoding

- Decoding with Translation Model and Language Model

<table>
<tr><th>เมื่อวาน</th><th>นี้</th><th>ฉัน</th><th>ไป</th><th>ทะเล</th><th>กับ</th><th>เพื่อน</th></tr>
<tr><td>Yesterday
(0.5)</td><td>this
(1.0)</td><td>I
(0.8)</td><td>go
(0.6)</td><td>the sea
(0.7)</td><td>with
(0.8)</td><td>friends
(0.6)</td></tr>
<tr><td colspan="2">Yesterday
(0.7)</td><td>I
(0.8)</td><td>went to
(0.3)</td><td>sea
(0.2)</td><td>with
(0.8)</td><td>friend
(0.4)</td></tr>
<tr><td colspan="2">Previous day
(0.1)</td><td>me
(0.2)</td><td>get to
(0.01)</td><td>ocean
(0.05)</td><td colspan="2">with my friend
(0.1)</td></tr>
</table>

Score = 0.7 x 0.8 x 0.6 x 0.7 x 0.8 x 0.6 x
P_{LM}(“Yesterday I go the sea with friends”)

Decoding

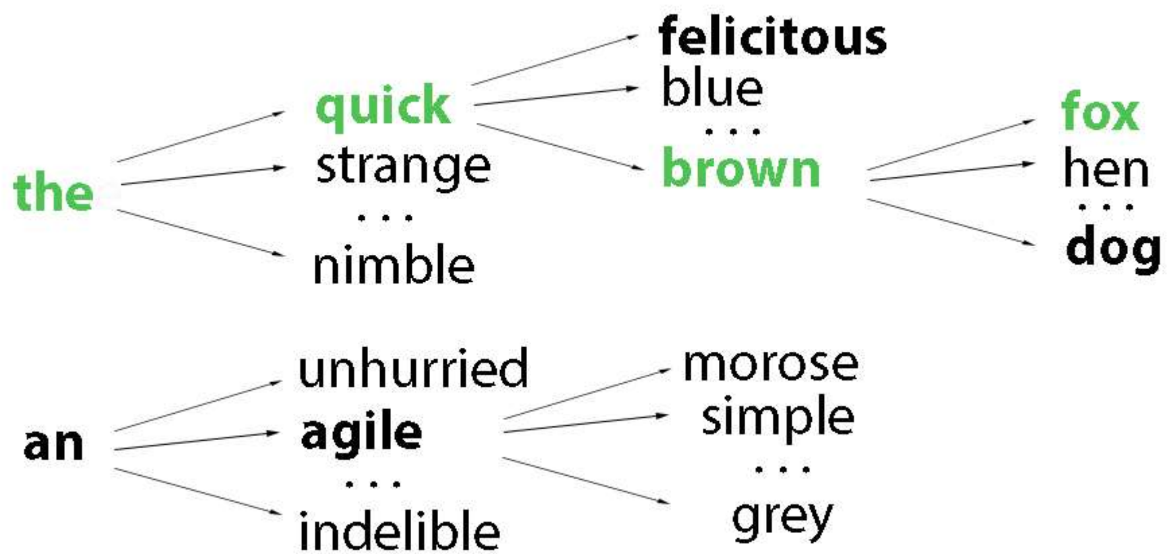
- Decoding with Translation Model and Language Model

<table>
<tr><th>เมื่อวาน</th><th>นี้</th><th>ฉัน</th><th>ไป</th><th>ทะเล</th><th>กับ</th><th>เพื่อน</th></tr>
<tr><td>Yesterday
(0.5)</td><td>this
(1.0)</td><td>I
(0.8)</td><td>go
(0.6)</td><td>the sea
(0.7)</td><td>with
(0.8)</td><td>friends
(0.6)</td></tr>
<tr><td colspan="2">Yesterday
(0.7)</td><td>I
(0.8)</td><td>went to
(0.3)</td><td>sea
(0.2)</td><td>with
(0.8)</td><td>friend
(0.4)</td></tr>
<tr><td colspan="2">Previous day
(0.1)</td><td>me
(0.2)</td><td>get to
(0.01)</td><td>ocean
(0.05)</td><td colspan="2">with my friend
(0.1)</td></tr>
</table>

Score = 0.1 x 0.2 x 0.3 x 0.05 x 0.1x
P_{LM}(“Previous day me went to ocean with my friend”)

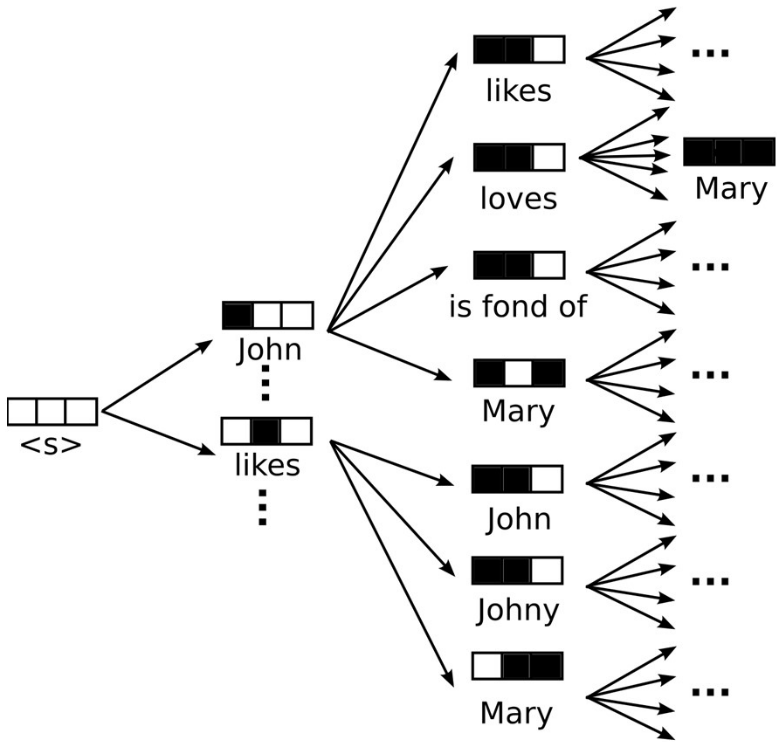
Beam Search

The Beam Search is a tree search algorithm but the data are filtered and sorted using a heuristic function.



Left-to-Right Beam Search

Beam Search



Beam Search in Statistical Decoder

SMT using NLTK Library

```
tmapi > tmapi > main_smt.py > test_smt
from nltk.translate import PhraseTable, StackDecoder
from collections import defaultdict
from math import log

def test_smt():
    phrase_table = PhraseTable()
    phrase_table.add(('book',), ('หนังสือ',), log(0.1))
    phrase_table.add(('this','book',), ('หนังสือ','เล่ม','นี้',), log(0.8))
    phrase_table.add(('this',), ('นี้',), log(0.8))
    phrase_table.add(('costs',), ('ราคา',), log(0.1))
    phrase_table.add(('300',), ('300',), log(0.1))
    phrase_table.add(('300',), ('สาม','ร้อย',), log(0.5))
    phrase_table.add(('baht',), ('บาท',), log(0.8))

    language_prob = defaultdict(lambda: -999.0)
    language_prob[('เล่ม',)] = log(0.5)
    language_prob[('หนังสือ',)] = log(0.4)
    language_prob[('หนังสือ', 'เล่ม', 'นี้')] = log(0.7)
    language_prob[('บาท',)] = log(0.1)
    language_prob[('สาม','ร้อย',)] = log(0.7)
    language_model = type('',(object,),
                          {'probability_change': lambda self, context, phrase: language_prob[ph
                           'probability': lambda self, phrase: language_prob[phrase]})()

    stack_decoder = StackDecoder(phrase_table, language_model)

    out = stack_decoder.translate("this book costs 300 baht".split())
    print(out)
```

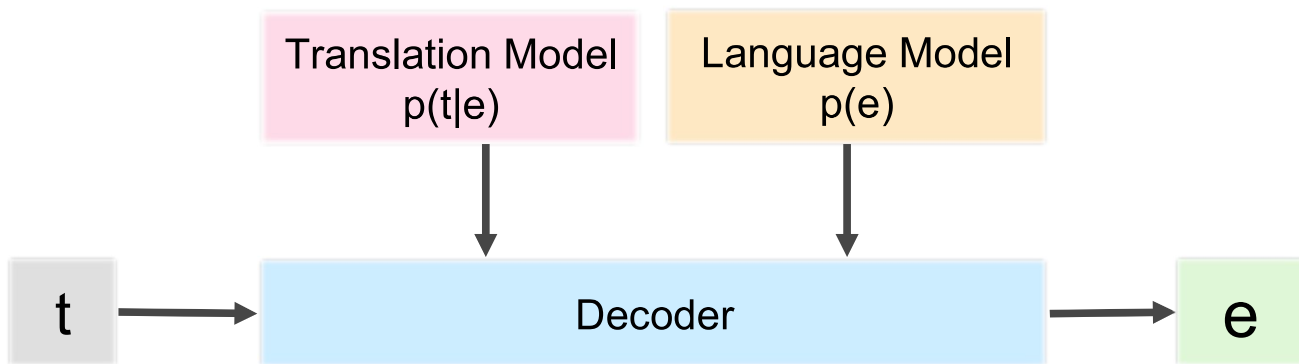
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
['หนังสือ', 'เล่ม', 'นี้', 'ราคา', 'สาม', 'ร้อย', 'บาท']
```

REF: https://www.nltk.org/api/nltk.translate.stack_decoder.html

Summary

- Statistical Machine Translation
- Translation Model
- Language Model
- Decoder



SMT Demo