

Statistical Machine Translation

Loïc Barrault

Loic.Barrault@univ-lemans.fr

Laboratoire d'Informatique de l'Université du Maine

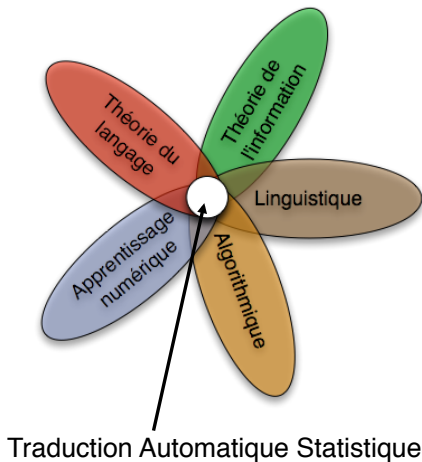
Plan

- Principle of statistical approach
- Translation model
 - Alignment
 - Phrase pair extraction
 - Probability estimation
- Quick recap of language modelling
- Decoding

Motivations

- Machine Translation = decision making
 - lexical choices
 - choice of order of words
 - choices of a particular expression given the context
 - etc.
- Statistical approach aims at taking decision based on several **models**

Statistical MT



Approaches

Approaches for MT

- Word by word
- Syntactic transfer
- Example-based
- Statistical MT

Approaches

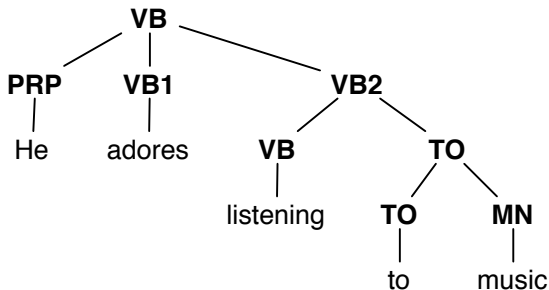
Translating word by word

- Use of a bilingual dictionary
 - Pros.:
 - simple to implement
 - provide the general idea of a text
 - Cons.:
 - (very) low translation quality
 - ex: word order
- not a viable solution to the problem!

Approaches

Syntactic transfer

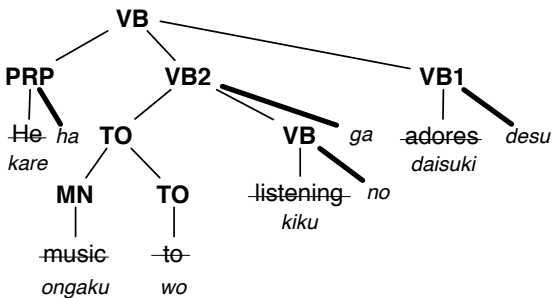
- Analysis of the sentence to obtain the constituents
- Reordering of those constituents
- Translation of the words (leaves of the tree)



Approaches

Syntactic transfer

- Analysis of the sentence to obtain the constituents
- Reordering of those constituents
- Translation of the words (leaves of the tree)



Approaches

Syntactic transfert

- Pros.
 - allows to reorder the words
 - ... taking into account the specificities of the language pair
- Cons.
 - needs a syntactic analyser susceptible to make errors
 - transfer rules are specific to the language pair
 - weak generalisation

Approaches

Example-based Machine Translation

- Fundamental idea:
 - Professional translator do not **always** translate using a deep analysis of the sentence
 - the sentence is decomposed into fragments which are translated separately and then recombined correctly
- Principle: translation by analogy
- cf. <https://www.linguee.com>

Issues:

- Localise the similar fragments
- Perform a sub-sentence alignment
- Combine **correctly** those fragments
- Select the best translation among multiple ones

Approaches

Example

- Translate "He buys a book on machine translation"
 - Examples in data base:
 - He buys an apple : Il achète une pomme
 - I read a book on statistics : Je lis un livre sur les statistiques
 - machine translation is great! : la traduction automatique, c'est génial !
-
- Pros.
 - reuse translation fragments generated by human translators (good quality)
 - Cons.
 - coverage limited by the amount of available data
 - dependent on the flexibility of the alignment algorithm

Approaches

Statistical Machine Translation

- Find the most probable sentence in target language given a sentence in source language
 - Automatic alignment of words and sentences of a bilingual corpus (bitext)
 - Probability estimation calculated from the bitext to build the translation model
- We will detail this approach in this course!

Principle of Statistical Machine Translation

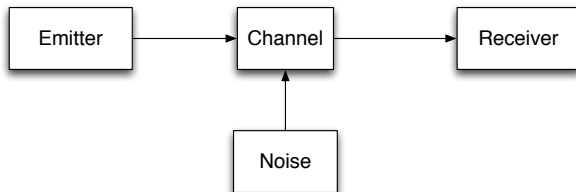
Noisy channel

- When I look at an article in Russian, I say: “This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode.”
- Warren Weaver (1949)



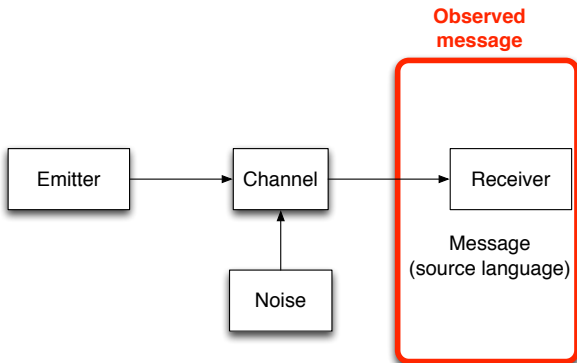
Noisy channel

- Theory of communication [Shannon, 1948]



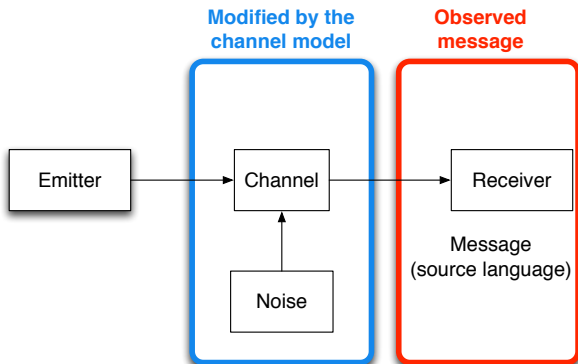
Noisy channel

- Theory of communication [Shannon, 1948]



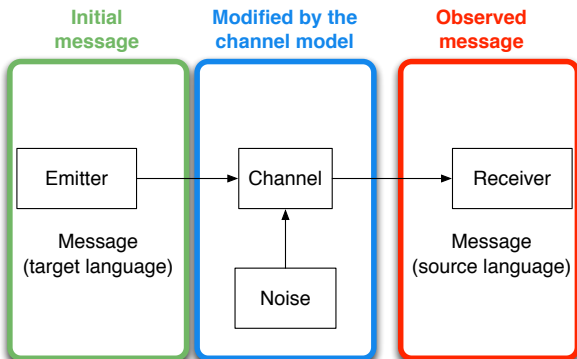
Noisy channel

- Theory of communication [Shannon, 1948]



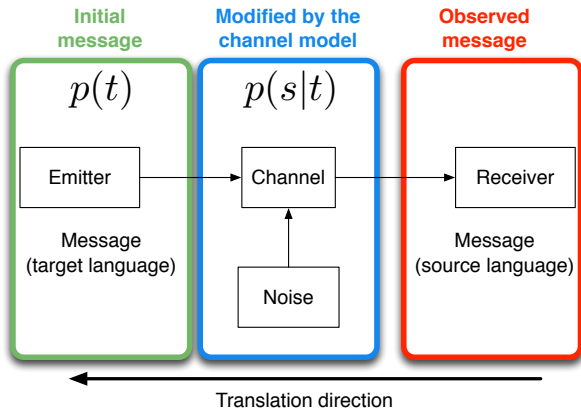
Noisy channel

- Theory of communication [Shannon, 1948]



Noisy channel

- Theory of communication [Shannon, 1948]



Problem formulation

- Translating a sentence = select among the **possible** translations the **best formed** sentence
- Translate sentence in source language s into a sentence in target language t

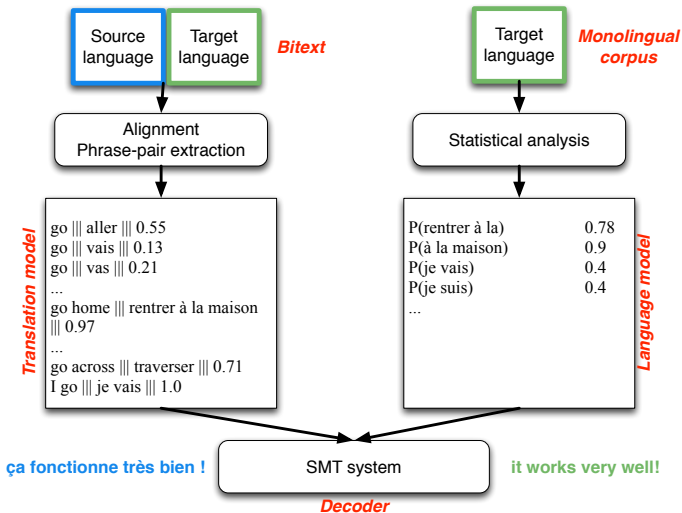
$$t^* = \operatorname{argmax}_t p(t|s) = \operatorname{argmax}_t p(s|t)p(t)$$

- $p(s|t) \Rightarrow$ translation model \Rightarrow Phrase Table (PT)
- $p(t) \Rightarrow$ statistical Language Model (LM)
- $\operatorname{argmax} \Rightarrow$ statistical decision making \Rightarrow Decoder

SMT principle

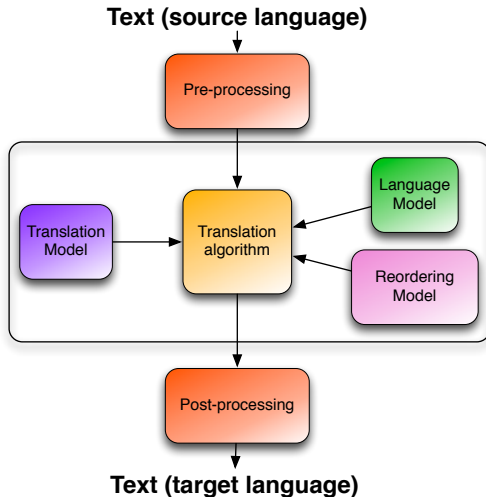
- Why not directly modelling $p(t|s)$?
 - 1 complex problem \rightarrow 2 easier sub-problems
 - TM and LM are cooperating
 - $p(s|t)$ is high for sentences having the correct words at the right place (roughly)
 - no matter if that sentence is well written in target language or not
 - $p(t)$ is high for well constructed sentences in target language
 - no matter if that sentence is a good translation of the source sentence or not
- \rightarrow **Finally, we get a well written sentence which is a good translation of the source sentence**

Statistical Machine Translation system



Translation Model

General architecture of an SMT system



Translation model

Statistical formulation

$$t^* = \operatorname{argmax}_t p(t|s) = \operatorname{argmax}_t p(s|t)p(t)$$

Issues

- How to estimate $p(s|t)$ for all sequences of words t
→ relative frequencies: $p(s|t) = \frac{C(s, t)}{C(t)}$
 - Impossible!
 - s and t are long sentences, they are **infrequent**
 - probability estimates by relative frequency is weak and irrelevant
- we have to decompose!

Decomposing $p(s|t)$

- s and t are decomposed in **phrases** of one or more words
- each **phrase** is translated separately
- target sentence is composed phrase after phrase
- reordering of phrases is possible
- translation probability is summed over all possible alignments:

$$\bullet p(s|t) = \sum_a p(a, s|t)$$

Step 1: alignment

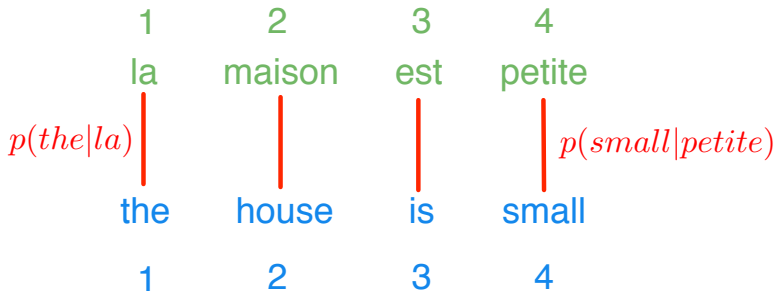
Word based alignment

- Translation unit is the word
- Translation model becomes:

- $$p(a, s|t) = \prod_{i=1}^m \text{lex}(s_i|t_j)$$

- with $\text{lex}(s_i|t_j)$ the probability that the word s_i translates into word t_j , named *lexical probability*
- We need a probabilistic dictionary!
 - contains all translations for a word
 - with their probabilities

Simple alignment example



- Alignment: $1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 4$

Lexical probabilities

Principle: analyse a text corpus

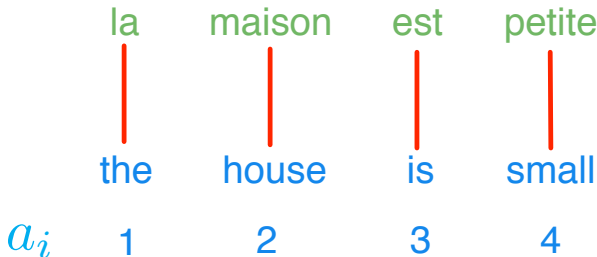
- Example: French word *maison*

Translation	# occurrences	probability
house	670	0.67
building	60	0.06
home	254	0.254
household	15	0.015
shell	1	0.001
Total	1000	1

- Probabilities calculated by **relative frequency**
- ⇒ Require a **word aligned** corpus!

Word alignment: formulation

- Let $s = s_1^m = s_1 \dots s_i \dots s_m$ be a sentence in source language
 - Let $t = t_1^n = t_1 \dots t_j \dots t_n$ be a sentence in target language
 - Let's define $a_1^n = a_1 \dots a_n$ with $a_j = i$ for $i = 1 \dots m, j = 1 \dots n$ as the alignment between s and t
- word j in target sentence is aligned with word i in source sentence



Word alignment

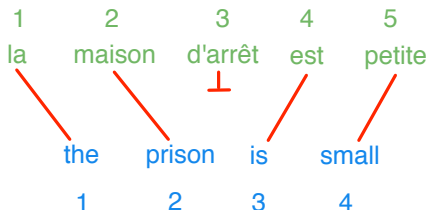
- Alignment 1 \rightarrow N



- 1 word in source language can translate into several words in target language

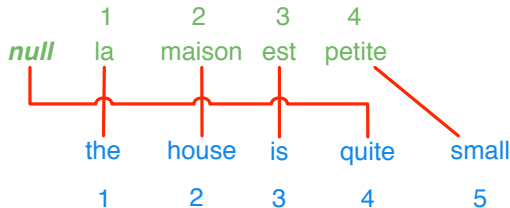
Word alignment

- Alignment $N \rightarrow 1$



- Le mathematical formulation cannot express this
- Source word not aligned to any word in target language
- Other examples:
- FR ne ... pas → [EN] not
- EN will learn → [FR] apprendra

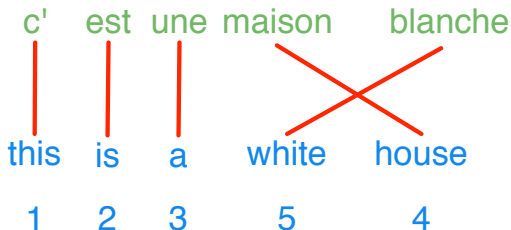
Word alignment



- Some words in target language may have no aligned words in source language

→ Add a special word ***null***

Word alignment



- Links can cross \Rightarrow reordering

- Examples:

FR noun adjective \rightarrow [EN] adjective noun

- verb adverb \rightarrow adverb verb
- complex reordering \rightarrow in [DE], the verb can be put at the end of the sentence

Word alignment: IBM model 1

- Lexical translations (word based)
- Data:
 - a source sentence $s = (s_1 \dots s_i \dots s_m)$, length is m
 - a target sentence $t = (t_1 \dots t_j \dots t_n)$, length is n
 - alignment function $a : j \rightarrow i$

$$p(t, a|s) = \frac{\epsilon}{(m+1)^n} \prod_{j=1}^n \text{lex}(t_j | s_{a(j)})$$

IBM model 1: example

la		maison		est		petite	
c	$\text{lex}(c s)$	c	$\text{lex}(c s)$	c	$\text{lex}(c s)$	c	$\text{lex}(c s)$
the	0.75	house	0.8	is	0.7	small	0.4
that	0.15	building	0.16	's	0.26	little	0.4
which	0.045	home	0.02	exists	0.025	short	0.1
who	0.03	household	0.015	has	0.01	minor	0.06
this	0.025	shell	0.005	are	0.005	pretty	0.04

$$\begin{aligned}
 p(t, a|s) &= \frac{\epsilon}{5^4} \times \text{lex}(\text{the}|\text{la}) \times \text{lex}(\text{house}|\text{maison}) \times \text{lex}(\text{is}|\text{est}) \dots \\
 &\quad \dots \times \text{lex}(\text{small}|\text{petite}) \\
 &= \frac{\epsilon}{5^4} \times 0.75 \times 0.8 \times 0.7 \times 0.4 \\
 &= 0.0002688\epsilon
 \end{aligned}$$

→ We need the **lexical probabilities**!

How to learn lexical probabilities?

- From a parallel corpus
- ⇒ but the word alignments are missing!

Chicken and egg problem

- With the alignments → estimate the lexical probabilities
 - With the lexical probabilities → calculate the alignments
- ⇒ Expectation - Maximization (EM) algorithm to the rescue!

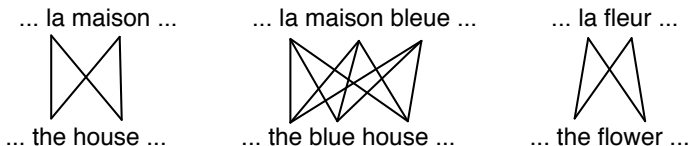
EM algorithm

- Expectation / Maximization
 - Solving incomplete data problem → alignments are missing in our case

Principle

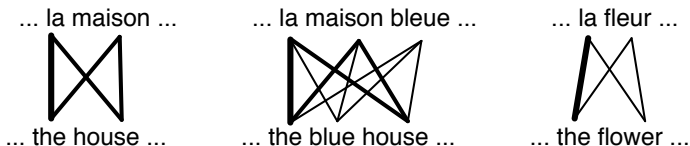
- ➊ Initialise model parameters (e.g. uniformly)
- ➋ Calculate probabilities of missing data (alignments)
→ Expectation
- ➌ Estimate model parameters with **completed data**
→ Maximization
- ➍ Repeat steps 2 and 3 until convergence (given a criterion)

EM algorithm



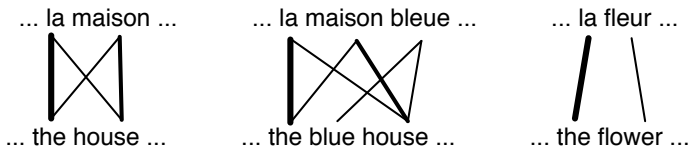
- Initial step: all alignments have same probability
 - Model learns:
 - **la** is often aligned with **the**
 - **maison** is often aligned with **house**
- need of large bilingual corpora (bitexts)

EM algorithm



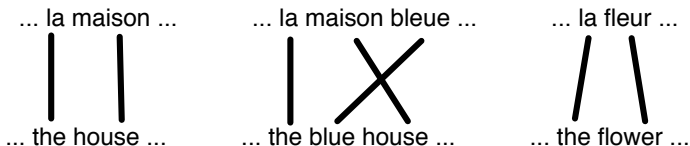
- After 1 iteration
- Alignments:
 - between **la** and **the** is more probable
 - same for **maison** and **house**
 - but also **la** and **house**, and **maison** and **the**

EM algorithm



- After another iteration
- Alignments between **fleur** and **flower** are more probable
→ Pigeonhole principle (*tiroirs de Dirichlet*)

EM algorithm



- After convergence
 - Hidden structure is revealed
- the final model parameters can now be calculated
- $p(la|the), p(maison|house), etc.$

IBM1 model and EM

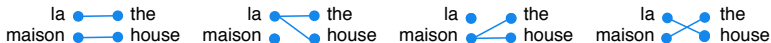
- Expectation: apply model on the incomplete data
 - give probabilities to alignments
 - given the current knowledge (model parameters)
- Maximization: estimate new model from completed data
 - collect counts (weighted by probabilities)
 - estimate new model parameters from those counts
- Repeat until convergence

IBM1 model and EM

- Probabilities:

- $p(the|la) = 0.7, p(house|la) = 0.05, p(the|maison) = 0.1,$
 $p(house|maison) = 0.8$

- Alignments:



$p(t,a s) = 0.56$	$p(t,a s) = 0.035$	$p(t,a s) = 0.08$	$p(t,a s) = 0.005$
$p(a s,t) = 0.824$	$p(a s,t) = 0.052$	$p(a s,t) = 0.118$	$p(a s,t) = 0.007$

- Counts:

$\text{count}(\text{the} la) = 0.824 + 0.052$	$\text{count}(\text{house} la) = 0.052 + 0.007$
$\text{count}(\text{the} maison) = 0.11 + 0.007$	$\text{count}(\text{house} maison) = 0.824 + 0.118$

Expectation

- We want to compute: $p(a|s, t)$
- Bayes rule:

$$p(a|s, t) = \frac{p(t, a|s)}{p(t|s)}$$

- Numerator is known \rightarrow IBM1 model
- Denominator remains to be computed

Expectation

$$\begin{aligned}
 p(t|s) &= \sum_a p(t, a|s) \\
 &= \sum_{a(1)=0}^m \cdots \sum_{a(n)=0}^m \frac{\epsilon}{(m+1)^n} \prod_{j=1}^n \text{lex}(t_j | s_{a(j)}) \\
 &= \frac{\epsilon}{(m+1)^n} \prod_{j=1}^n \sum_{i=0}^m \text{lex}(t_j | s_i)
 \end{aligned}$$

- Note: the trick in last line makes it tractable (suppress exponential number of products)

Expectation

- Putting everything together:

$$\begin{aligned}
 p(a|s, t) &= p(t, a|s)/p(t|s) \\
 &= \frac{\frac{\epsilon}{(m+1)^n} \prod_{j=1}^n \text{lex}(t_j|s_{a(j)})}{\frac{\epsilon}{(m+1)^n} \prod_{j=1}^n \sum_{i=0}^m \text{lex}(t_j|s_i)} \\
 &= \prod_{j=1}^n \frac{\text{lex}(t_j|s_{a(j)})}{\sum_{i=0}^m \text{lex}(t_j|s_i)}
 \end{aligned}$$

Maximization

- Collect the counts:

$$\text{count}(t|s; S, T) = \sum_a p(a|s, t) \sum_{j=1}^n \delta(T, t_j) \delta(S, s_{a(j)})$$

with $\delta(C, c) = 1$ if $c \in C$ else 0

- using the same trick as before:

$$\text{count}(t|s; S, T) = \frac{\text{lex}(t|s)}{\sum_{i=0}^m \text{lex}(t|s_i)} \sum_{j=1}^n \delta(T, t_j) \sum_{i=0}^n \delta(S, s_i)$$

Maximization

- Now we can estimate the model


$$p(t|s; T, S) = \frac{\sum_{(S,T)} count(t|s; S, T)}{\sum_s \sum_{(S,T)} count(t|s; S, T)}$$


Pseudocode


Algorithm 1 IBM1 model and EM : pseudocode

1: Input : a bilingual corpus (S,T)	16: // collect counts
2: Output : translation prob. $p(t s)$	17: for all words $t \in T$ do
3: initialise $p(t s)$ uniformly	18: for all words $s \in S$ do
4: while not converged do	19: $count(t s) + = \frac{t(t s)}{t-total(t)}$
5: // initialisation	20: $total(s) + = \frac{t(t s)}{t-total(t)}$
6: $count(t s) = 0$ for all s, t	21: end for
7: $total(s) = 0$ for all s	22: end for
8: for all sentence pairs (S,T) do	23: end for // $\forall (S, T)$
9: // compute normalisation	24: // Estimate probabilities
10: for all words $t \in T$ do	25: for all source words s do
11: $t-total(t) = 0$	26: for all target words t do
12: for all words $s \in S$ do	27: $p(t s) = \frac{count(t s)}{total(s)}$
13: $t-total(t) + = p(t s)$	28: end for
14: end for	29: end for
15: end for	

Convergence - example

das Haus

 the house

das Buch

 the book

ein Buch

 a book

<i>e</i>	<i>f</i>	initial	1st it.	2nd it.	3rd it.	...	final
the	das	0.25	0.5	0.6364	0.7479	...	1
book	das	0.25	0.25	0.1818	0.1208	...	0
house	das	0.25	0.25	0.1818	0.1313	...	0
the	buch	0.25	0.25	0.1818	0.1208	...	0
book	buch	0.25	0.5	0.6364	0.7479	...	1
a	buch	0.25	0.25	0.1818	0.1313	...	0
book	ein	0.25	0.5	0.4286	0.3466	...	0
a	ein	0.25	0.5	0.5714	0.6534	...	1
the	haus	0.25	0.5	0.4286	0.3466	...	0
house	haus	0.25	0.5	0.5714	0.6534	...	1

- Source: MTMarathon lectures

→ <http://lium3.univ-lemans.fr/mtmarathon2010/programme.php>

Perplexity

- How much does the model matches with the data?
- Perplexity:

$$\log_2 PP = - \sum_{i,j} \log_2 p(t_j | s_a(i))$$

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

IBM models

- IBM1: lexical translation
 - IBM2: + absolute reordering model
 - IBM3: + fertility model
 - IBM4: + relative reordering model
 - IBM5: fixes deficiency
- everything builds on IBM1 model!

HMM model

- Words do not move independently of each other
- Often move in groups

→ condition word movements on previous word

- HMM alignment model

$$p(a(j)|ja(j-1), m)$$

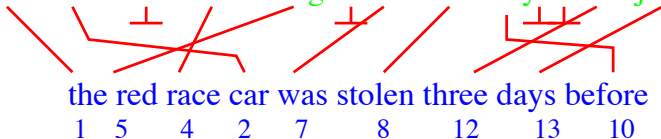
- EM algorithm implementation harder (requires dynamic programming)

Step 2: phrase pairs extraction

More complex alignment

- Things can quickly get messy!

la voiture de course rouge a été volée il y a trois jours .



- Limitation of word alignment

More complex alignment

- That's better!



- Limitation of word alignment
- Would be better by aligning **group of words** \Rightarrow *phrases*
- Simpler, more realistic!

Phrase alignment

- Let \tilde{s} and \tilde{t} be two sequences of words, respectively in source and target language
- When we know the alignment, we can compute:

$$p(\tilde{s}_i|\tilde{t}_j) = \frac{C(\tilde{s}_i), \tilde{t}_j)}{C(\tilde{t}_i)}$$

- We have: parallel text aligned at sentence level (Europarl, U.N., etc.)
 - BUT: costly and hard to create an manual alignment
- ⇒ create this alignment **automatically**

Alignment in both directions

- Perform the alignment in two directions



- Remember: alignment is not **symmetrical** $\Rightarrow p(s|t) \neq p(t|s)$

Step 2a: alignment matrix

before													
days													
three													
stolen													
was													
car													
race													
red													
the													
	la	voiture	de	course	rouge	a	été	volée	il	y	a	trois	jours

Step 2a: alignment matrix

before									X				
days													X
three												X	
stolen								X					
was							X						
car		X											
race				X									
red					X								
the	X												
	la	voiture	de	course	rouge	a	été	volée	il	y	a	trois	jours

- Mark word alignments which are compatible with symmetrized alignments

Step 2a: alignment matrix

before									X				
days												X	
three											X		
stolen							X						
was							X						
car		X											
race				X									
red					X								
the	X												
	la	voiture	de	course	rouge	a	été	volée	il	y	a	trois	jours

- group words, maximum size 2

Step 2a: alignment matrix

before									X				
days													X
three												X	
stolen								X					
was							X						
car		X											
race				X									
red					X								
the	X												
	la	voiture	de	course	rouge	a	été	volée	il	y	a	trois	jours

- Unauthorized group

Step 2a: alignment matrix

before									X				
days													X
three											X		
stolen							X						
was							X						
car		X											
race				X									
red					X								
the	X												
	la	voiture	de	course	rouge	a	été	volée	il	y	a	trois	jours

- Extending groups

Step 2a: alignment matrix

before									X				
days												X	
three											X		
stolen								X					
was							X						
car		X											
race				X									
red					X								
the	X												
	la	voiture	de	course	rouge	a	été	volée	il	y	a	trois	jours

- group of words, maximum size 2

Step 2a: alignment matrix

before									X				
days													X
three											X		
stolen								X					
was							X						
car		X											
race				X									
red					X								
the	X												
	la	voiture	de	course	rouge	a	été	volée	il	y	a	trois	jours

- group of words, maximum size 3

Step 2a: alignment matrix

before									X				
days													X
three												X	
stolen								X					
was							X						
car		X											
race				X									
red					X								
the	X												
	la	voiture	de	course	rouge	a	été	volée	il	y	a	trois	jours

- group of words, maximum size 4

Step 2a: alignment matrix

before									X				
days													X
three												X	
stolen								X					
was							X						
car		X											
race				X									
red					X								
the	X												
	la	voiture	de	course	rouge	a	été	volée	il	y	a	trois	jours

- group of words, maximum size 5

Step 3: build phrase-table

Step 3: build phrase table

Source s	# Target t	# occurrences	$p(\tilde{t} \tilde{s})$	$p(\tilde{s} \tilde{t})$
I go	je vais	1237	0.615	0.844
I go	je marche	75	0.375	0.434
I go	je rentre	2	0.01	...
I run	je vais	23	0.192	0.157
I run	je cours	97	0.808	...
I walk	je marche	98	0.98	0.566
...

- Use of inverse probabilities → different because alignments are not symmetrical

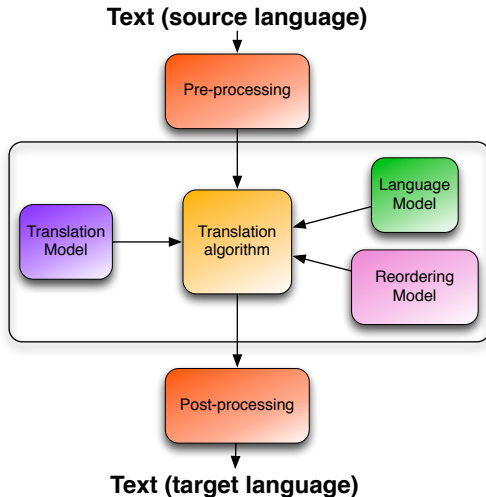
Phrase table in practice

- 5 scores are stored in it
 - $p(\tilde{s}|\tilde{t})$: phrase translation probability, $t \rightarrow s$
 - $lex(\tilde{s}|\tilde{t})$: lexical phrase translation probability, $t \rightarrow s$
 - $p(\tilde{t}|\tilde{s})$: phrase translation probability, $s \rightarrow t$
 - $lex(\tilde{t}|\tilde{s})$: lexical phrase translation probability, $s \rightarrow t$
 - phrase-pair insertion penalty $\rightarrow e^1 \approx 2.718$
- with:

$$lex(\tilde{s}|\tilde{t}) = \prod_i lex(s_{a(i)}|t_i)$$

Step 4: Quick recap of Language Modelling

General architecture of an SMT system



Language Model - LM

- One of the main component of the system
- Allows to distinguish between well written sentences and bad ones
- Should give priority to grammatically and semantically correct sentences
 - in a implicit fashion, no need for a syntactic nor semantic analysis
 - Monolingual process → no adequacy with source sentence here

Language Model - LM

- Goal: provide a non zero probability to **all** sequences of words
 - even for non-grammatical sentences
 - learned automatically from texts

Issues:

- How to assign a probability to unseen sequences?
- How to manage the model size?
- How to ensure good probability estimates?

Language Model

- Goal: provide a non zero probability to all sequences of words W extracted from a **vocabulary** V
- Vocabulary: list of all words known by the model
 - a specific word **<unk>** to manage all the words not in V
 - word = sequence of characters without space
 - word \neq linguistic word \rightarrow token

Let $W = (w_1, w_2, \dots, w_n)$ with $w_i \in V$ be a word sequence

$$p(W) = \prod_{i=1}^T p(w_i | h_i)$$

with $h_i = (w_1, w_2, \dots, w_{i-1})$ the history of word w_i

LM - Complexity

- Complexity for a vocabulary size of 65k
 - $65k^2 = 4\,225\,000\,000$ sequences of 2 words
 - $65k^3 = 2.74 \times 10^{14}$ sequences of 3 words

- Equivalence classes

- group histories in equivalence classes ϕ

$$p(W) \approx \prod_{i=1}^T p(w_i | \phi(h_i))$$

- Language modelling lies in determining ϕ and find a method to estimate the corresponding probabilities
- see work by Frederick Jelinek

LM - n-gram

- n -gram: sequence of n words
- n -gram model uses an equivalence class mapping the history h_i to the $n - 1$ previous words
- bigram model: $\phi(h_i) = (w_{i-1})$

$$p(W) \approx p(w_1) \times \prod_{i=2}^T p(w_i | w_{i-1})$$

- trigram model: $\phi(h_i) = (w_{i-1}, w_{i-2})$

$$p(W) \approx p(w_1) \times p(w_2 | w_1) \times \prod_{i=3}^T p(w_i | w_{i-1}, w_{i-2})$$

- n -gram: $\phi(h_i) = (w_{i-1}, \dots, w_{i-n+1})$
- Consequences:
 - $n - 1$ words are enough to predict the next word
 - in practice: n ranges to 4 or 5, barely 6 \Rightarrow require exponential quantity of data

LM - Characteristics

- Language structure implicitly captured
 - probability of succeeding words, cooccurrences
 - same for semantic
- Probabilities are independent from the position in the sentence
 - add begin (<s>) and end (</s>) of sentence tokens
- Probabilities are estimated using a large quantity of data (corpus), which are supposed to be **well written**

LM - probabilities

- Unigram probabilities

$$p(w_i) = \frac{C(w_i)}{\sum_k C(w_k)} = \frac{C(w_i)}{\text{corpus size}}$$

→ $C(.)$ is the counting function

- n -gram probabilities

$$p(w_i|h_i^n) = \frac{C(h_i^n w_i)}{C(h_i^n)}$$

LM - Unseen sequences

- Sequences that are not allowed by the language
 - Ex.: "ils part tôt", "elle est beau"
- Sequences that are not seen in the training corpus

Solutions

- Increase training corpus size
- makes training longer + can we ever get a perfect corpus?
- Reserve a (small) probability mass to unseen events

$$\epsilon \geq p(w_i | h_i^n) > 0 \quad \forall i, \forall h$$

LM - Smoothing

- Idea: take probability mass D to seen events, then redistribute it to unseen events
- smoothing
- Going further: comparative study
- Stanley F. Chen and Joshua T. Goodman, *An Empirical Study of Smoothing Techniques for Language Modelling*. Computer, Speech and Language, 13(4), pp. 359-394, 1999.

LM - Backoff

- Idea: exploit lower order history
- Backoff technique

$$\tilde{p}(w_i|h_i^n) = \begin{cases} p^-(w_i|h_i^n) & \text{if } C(h_i^n w_i) > 0 \\ \alpha(h_i^n) p^-(w_i|h_i^{n-1}) & \text{if } C(h_i^n w_i) = 0 \end{cases}$$

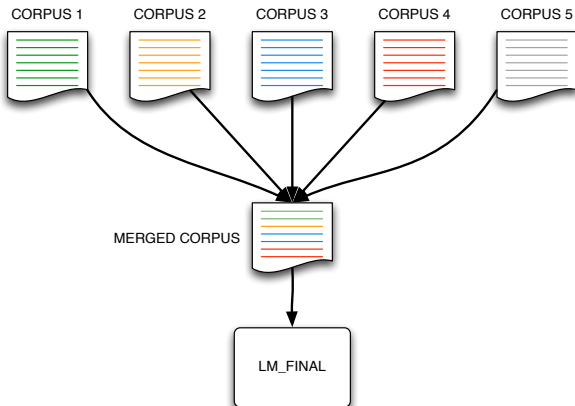
- with $\alpha(h_i^n)$ the backoff weight
- computed so that probability distribution is respected (probs between 0 and 1 and sums to 1)

LM - In practice for MT

- Vocabulary:
 - Use all the ~~words~~ tokens belonging to **in domain** corpora
 - target side of train and development corpora
 - specialized monolingual corpora
 - Most frequent ~~words~~ tokens of large generic corpora
 - seen at least k times

LM - Training methodology

- Merge training data, standard training procedure

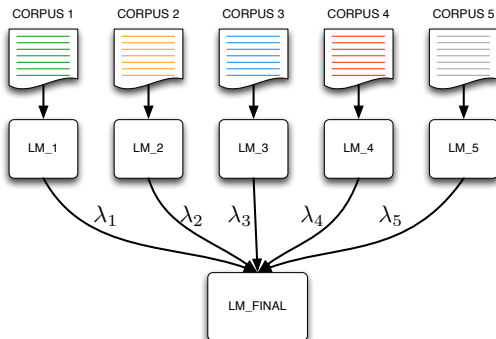


LM - Training methodology

- (log) linear Interpolation
- with J models:

$$p(w_i|h_i^n) = \sum_{j=0}^J \lambda_j \cdot p_j(w_i|h_i^n)$$

→ λ_j are computed using an EM procedure



LM - perplexity

- Perplexity: measure corresponding to the ability of the model to predict word of an unseen text
 - unseen = not in training dat
 - criterion to be minimized

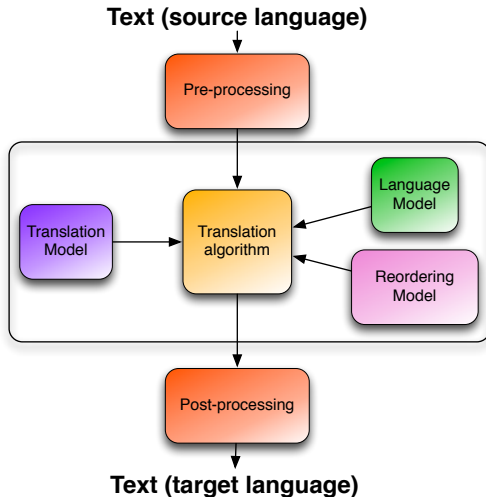
$$\begin{aligned}PPL &= 2^H \quad [H \text{ is the cross-entropy}] \\&= \log \prod_{i=1}^T p(w_i)^{-\frac{1}{T}} \\&= \sum_{i=1}^T \log p(w_i)^{-\frac{1}{T}} \quad [\text{the log sum}] \\&= -\frac{1}{T} \sum_{i=1}^T \log p(w_i) \quad [\text{normalisation by } T]\end{aligned}$$

LM - SRILM

- See [Stolcke, 2002]
 - Build a model: ngram-count
- !! always specify order with **-order N** and use of unknown class with **-unk**
- Compute interpolation weights: compute-best-mix
use the outputs of the following command: **ngram -debug 2 -ppl ...**
 - Compute perplexity, interpolated model: ngram
-ppl <dev corpus>: compute perplexity on development corpus
-mix-lmK <lmK> -mix-lambdaK <coeffK> : interpolate several models <lmK> with weights <coeffK> (K ranging from 0 to 9)

Step 5: Decoding

General architecture of an SMT system



Decoding

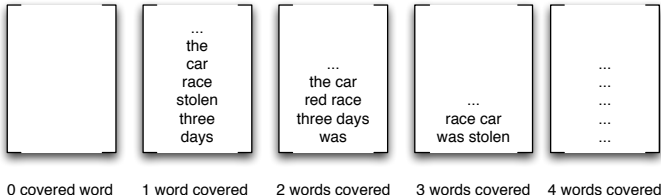
- Statistical formulation:

$$t^* = \operatorname{argmax}_t p(t|s) = \operatorname{argmax}_t p(s|t)p(t)$$

- $p(s|t)$ computed with the phrase-table
 - $p(t)$ with the language model
- We have to find t^* → this is the decoder's job

Decoding - Stack decoding

- Go through the source sentence
- Create translation hypotheses
 - 1st iteration: cover 1 source word
 - 2nd iteration: cover 2 source words
 - etc.
- Ex: *la voiture de course rouge a été volée il y a trois jours*



- Scores are propagated along with the hypotheses

Decoding

- NP-hard problem → exponential number of hypotheses
 - Solution: reduce the search space

Recombine partial hypotheses → no loss

- Hypotheses leading to the same state but with a lower score can be stopped
 - Ex.: *la voiture de course rouge a été volée il y a trois jours*
 - 1st hypo: "la voiture de course rouge" → "the red race car", using only 1 phrase pair (seen in phrase-table)
 - 2nd hypo: "la voiture de course rouge" → "the red race car", using 3 phrase pairs "la"→"the", "voiture de course"→"race car", "rouge"→"red"
- Source word coverage and partial hypotheses identical ⇒ keep only the one with best score

Absolute or relative pruning → with loss

Decoding

- NP-hard problem \rightarrow exponential number of hypotheses
 - Solution: reduce the search space

Recombine partial hypotheses \rightarrow no loss

Absolute or relative pruning \rightarrow with loss

- keep only k best partial hypotheses
- keep partial hypotheses having a score not less than $x\%$ of best score

Full system

SMT system - Weighted log linear model

- Initial formulation: $t^* = \operatorname{argmax}_t p(t|s) = \operatorname{argmax}_t p(s|t)p(t)$
- In reality:

$$p(t|s) = \exp \sum_{i=1}^{FF} \lambda_i ff_i(s, t)$$

- with ff_i : a feature function
- $p(s|t)$, $lex(s|t)$, $p(t|s)$, $lex(t|s)$, $plm(t)$, etc.
- Some features more important than others
 - Add a weight λ_i to each feature function ff_i
 - Those weights are optimised → see MERT (Minimum Error Rate Training)

SMT - Conclusion

Pros.

- Knowledge is automatically extracted from a large quantity of bilingual texts
- Human intervention is reduced to minimum → data creation remains the main bottleneck

Cons.

- Syntax is not dealt with explicitly
- Weak generalisation power
 - The system can translate correctly "10 red apples" and fail translating "11 red apples"
 - Knowledge is limited to what is encountered in the bitexts
- Inspection is complex: very difficult to predict the impact of a change

→ data, feature functions, etc.

SMT - Conclusion

- All tools are open source!
- Moses toolkit: [Koehn et al., 2007]
 - see <http://statmt.org/moses/>
 - see <https://github.com/moses-smt>
- Contains training, preprocessing and postprocessing scripts
- MERT: [Och, 2003]
 - provided with Moses
- Bilingual alignment:
 - GIZA++: [Gao and Vogel, 2008]
 - fast_align: [Dyer et al., 2013]
 - See http://www.cdec-decoder.org/guide/fast_align.html

References I

Dyer, C., Chahuneau, V., and Smith, N. A. (2013).
A simple, fast, and effective reparameterization of ibm model 2.
In *Proceedings of NAACL*.

Gao, Q. and Vogel, S. (2008).
Parallel implementations of word alignment tool.
In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, SETQA-NLP '08, pages 49–57.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007).
Moses: Open source toolkit for statistical machine translation.
In *Meeting of the Association for Computational Linguistics*, pages 177–180.

References II

Och, F. J. (2003).

Minimum error rate training in statistical machine translation.
In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 160–167, Stroudsburg, PA, USA. Association for Computational Linguistics.

Shannon, C. E. (1948).

A mathematical theory of communication.
Bell Systems Technical Journal, 27:379–423, 623–656.

Stolcke, A. (2002).

Srilm - an extensible language modeling toolkit.
Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP 2002), pages 901–904.