Traduction Automatique

Plan

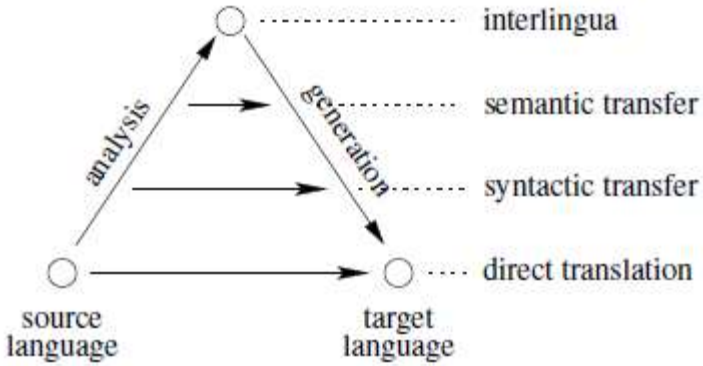
- Traduction Automatique à base de règles

- Traduction Automatique à base de corpus

  - Traduction à base d'exemples
  - Traduction statistique
  - Traduction neuronale

Traduction Automatique à base de règles

Approches à base de règles



Approche Interlangue

1. Représentation syntactico-sémantique interlangue du texte source
2. Génération du texte cible en partant de la représentation

Approche par transfert

1. Analyse lexicale et syntaxique du texte source
2. Transfert des structures et des traductions lexicales en langue cible

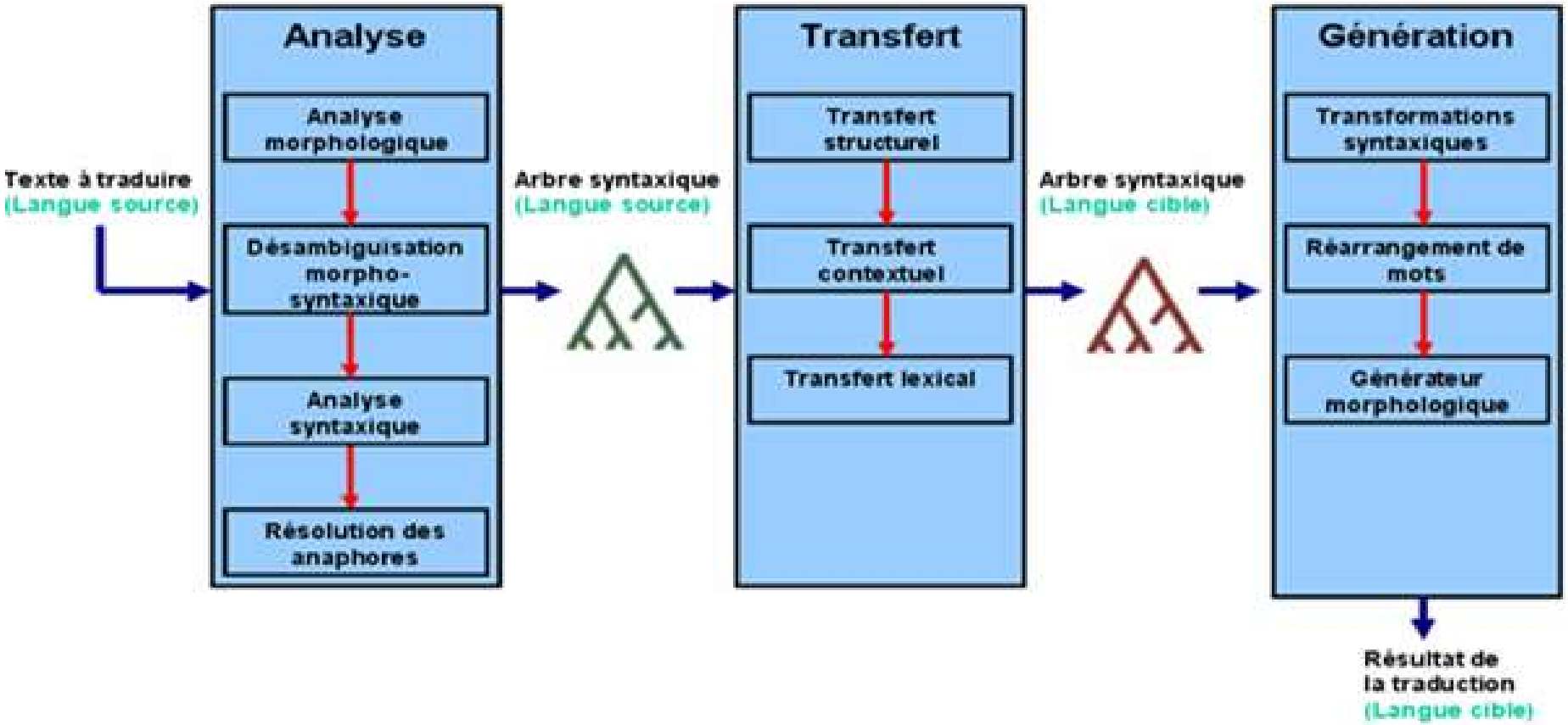
Approche directe

1. Traduction mot à mot du texte source vers le texte cible
2. Modification de l'ordre des mots traduits dans le texte cible

Approche directe

- Utilisation d'un dictionnaire bilingue
- Avantages :
  - simple à implémenter
  - donne l'idée générale du texte
- Inconvénients :
  - faible qualité de la traduction
  - ex: ordre de mots

Approche à base de transfert



Types de transfert

Transfert lexical

- Exemple : *car.Noun* ↔ *voiture.Noun*
- Besoins : dictionnaires bilingues

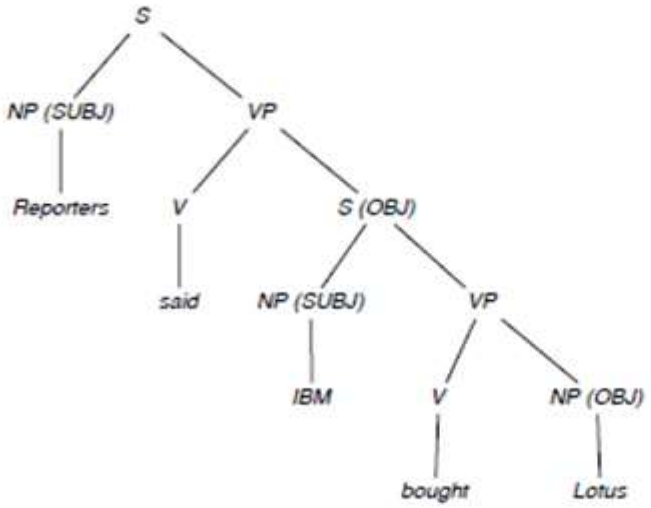
Transfert de structures

- Exemple : *the Adj Noun* ↔ *le Noun Adj+d*
- Besoins : analyseur syntaxique et formalisme de transfert (ex. transducteurs d'arbres)

Transfert lexico-syntaxique

- Exemple : *NP consist of NP* ↔ *NP consister en NP*
- Besoins : idem que pour les transferts structures + lexiques syntaxiques

Transfert syntaxique



- Analyse en constituants de la phrase
- Réarrangement de ces constituants
- Traduction des mots

Transfert syntaxique: Limites

- Avantages
  - permet de réordonner les mots
- Inconvénients
  - règles de transfert spécifiques à la paire de langues
  - nécessite un analyseur syntaxique susceptible de faire des erreurs

Traduction Automatique à base de corpus

- **Traduction à base d'exemples**
- **Traduction statistique**
- **Traduction neuronale**

Traduction basée sur les exemples

- Idée fondamentale :
  - Les traducteurs professionnels ne traduisent pas (toujours) en effectuant une analyse profonde de la phrase
  - La phrase est décomposée en fragments, ces fragments sont traduit puis recomposés correctement
- Principe : traduction par analogie

Traduction basée sur les exemples: Défis

- Traduire “He buys a book on machine translation”
- Exemples :
  - **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 !

Traduction basée sur les exemples: Limites

- Problématiques
  - Localiser les fragments similaires
  - Effectuer un alignement sous-phrastique
  - Combiner **correctement** les fragments obtenus
  - Sélectionner la meilleure traduction parmi un ensemble
- Avantages
  - réutilise des fragments de traductions issues de traducteurs humains (qualité)
- Inconvénients
  - couverture limitée selon la taille de la base de données et la flexibilité de l'algorithme de mise en correspondance

Traduction Statistique

Traduction Statistique: Principes

- Trouver la phrase en langue cible la plus probable étant donné une phrase en langue source
- Alignement automatique des mots et des phrases d'un bitexte (corpus parallèle)
- Estimation des probabilités à partir de ce bitexte afin de construire un modèle de traduction

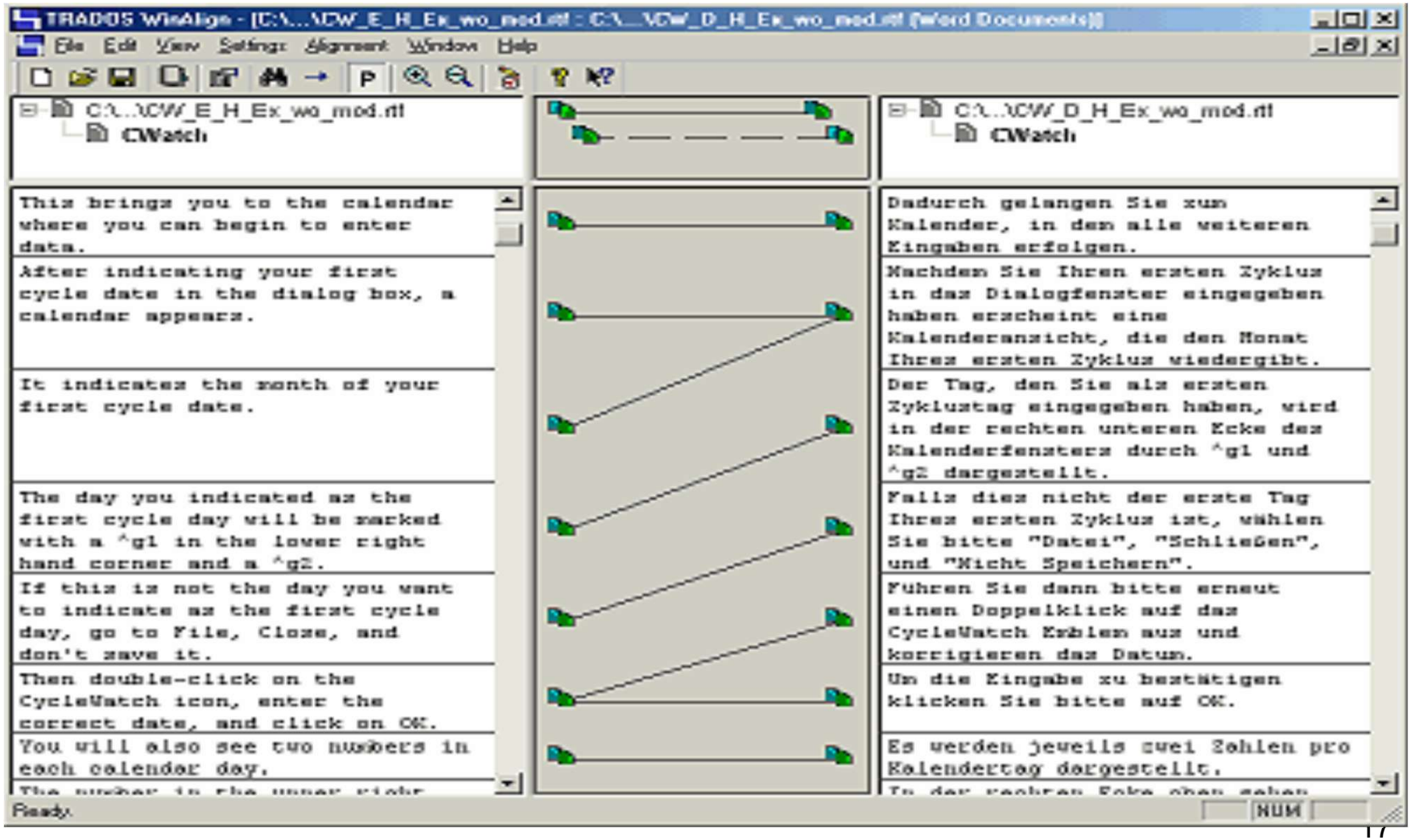
Traduction Statistique: Deux sources de connaissance

- Traductions humaines (bitextes)

- Textes en langue cible

GERMAN	ENGLISH	FRENCH
Einleitung	Introduction	Introduction
I. Von dem Unterschiede der reinen und empirischen Erkenntnis	*I. Of the difference between Pure and Empirical Knowledge*	*I. De la différence de la connaissance pure et de la connaissance empirique.*
Daß alle unsere Erkenntnis mit der Erfahrung anfange, daran ist gar kein Zweifel: denn wodurch sollte das Erkenntnisvermögen sonst zur Ausübung erweckt werden, geschähe es nicht durch Gegenstände, die unsere Sinne rühren und teils von selbst Vorstellungen bewirken, teils unsere Verstandestätigkeit in Bewegung bringen, diese zu vergleichen, sie zu verknüpfen oder zu trennen, und so den rohen Stoff sinnlicher Eindrücke zu einer Erkenntnis der Gegenstände zu verarbeiten, die Erfahrung heißt? Der Zeit nach geht also keine Erkenntnis in uns vor der Erfahrung vorher, und mit dieser fängt alle an.	That all our knowledge begins with experience there can be no doubt. For how is it possible that the faculty of cognition should be awakened into exercise otherwise than by means of objects which affect our senses, and partly of themselves produce representations, partly rouse our powers of understanding into activity, to compare to connect, or to separate these, and so to convert the raw material of our sensuous impressions into a knowledge of objects, which is called experience? In respect of time, therefore, no knowledge of ours is antecedent to experience, but begins with it.	Que toute notre connaissance commence avec l'expérience, cela ne soulève aucun doute. En effet, par quoi notre pouvoir de connaître pourrait-il être éveillé et mis en action, si ce n'est par des objets qui frappent nos sens et qui, d'une part, produisent par eux-mêmes des représentations et, d'autre part, mettent en mouvement notre faculté intellectuelle, afin qu'elle compare, lie ou sépare ces représentations, et travaille ainsi la matière brute des impressions sensibles pour en tirer une connaissance des objets, celle qu'on nomme l'expérience? Ainsi, chronologiquement, aucune connaissance ne précède en nous l'expérience et c'est avec elle que toutes commencent.

Traduction Statistique: Construction de corpus alignés phrase à phrase

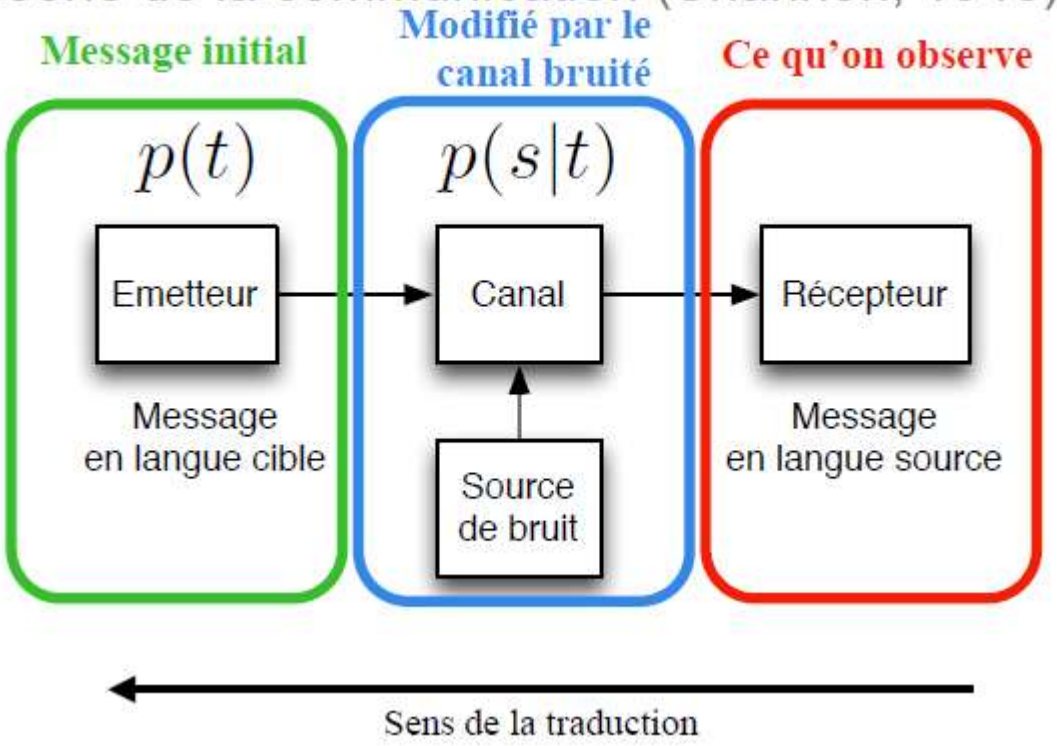


Traduction Statistique: Canal bruité

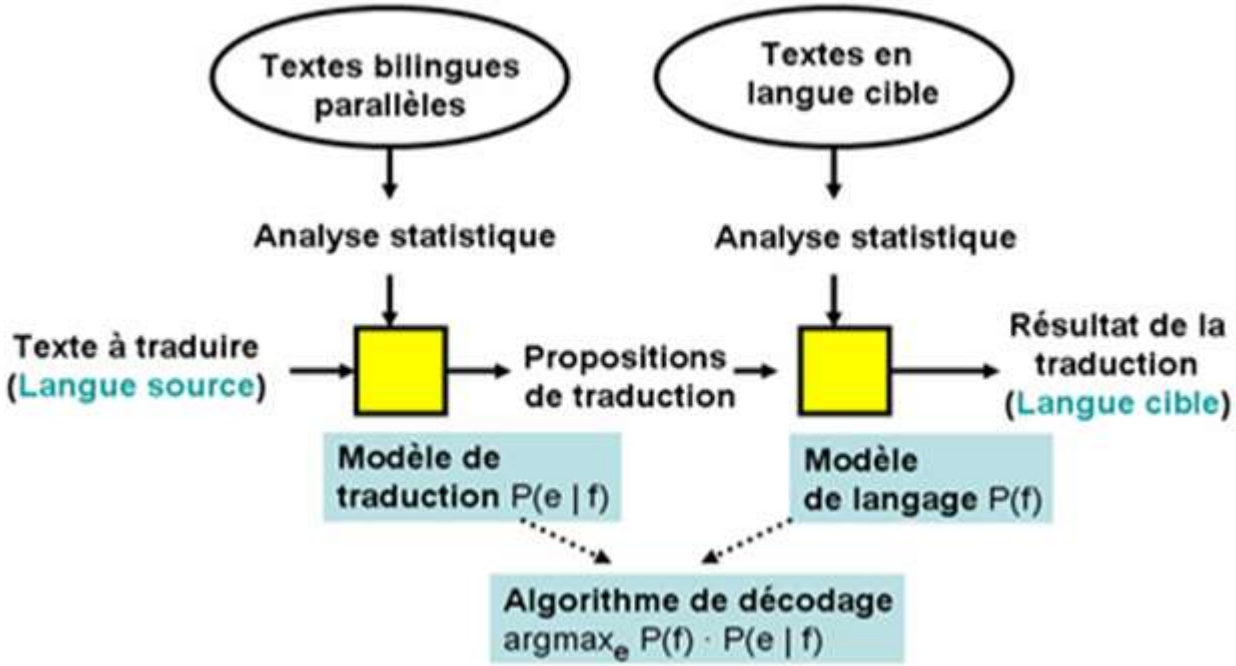
- When I look at an article in Russian, I say: "This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode."
- Warren Weaver (1949)

Traduction Statistique: Canal bruité

- Théorie de la communication (Shannon, 1948)



Traduction Statistique: Formulation du problème



Traduction Statistique: Formulation du problème

- Traduire une phrase = choisir la phrase la mieux formée parmi les traductions possibles
- Traduction d'une phrase en langue source ***s*** vers une langue cible ***t***

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

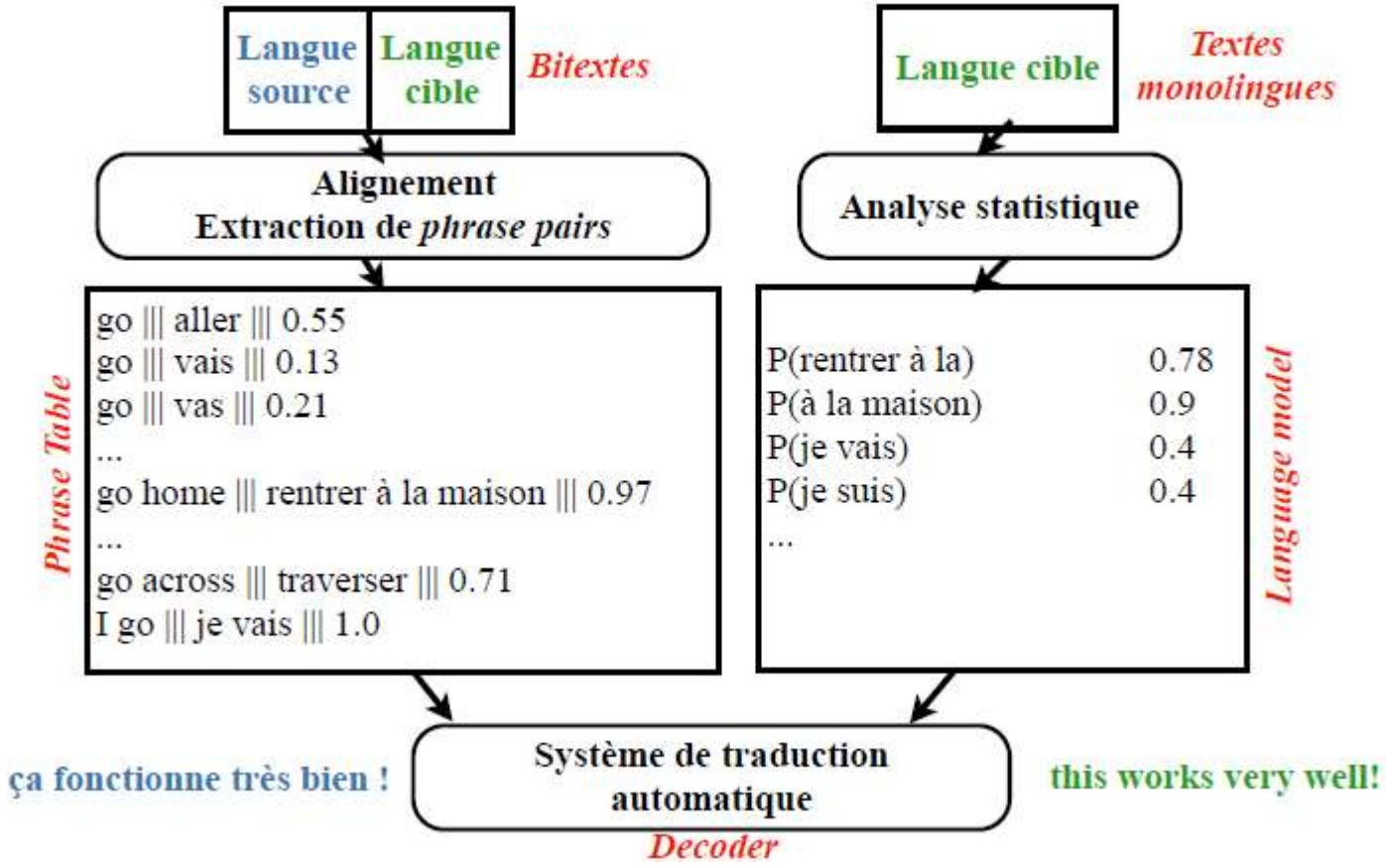
$$= \underset{t}{\operatorname{argmax}}\ p(s|t)\ p(t) \quad \text{(Bayes)}$$

  - Modèle de traduction - Phrase Table (PT)
  - Modèle de langage statistique - Language Model (LM)
  - Prise de décision statistique - Decoder

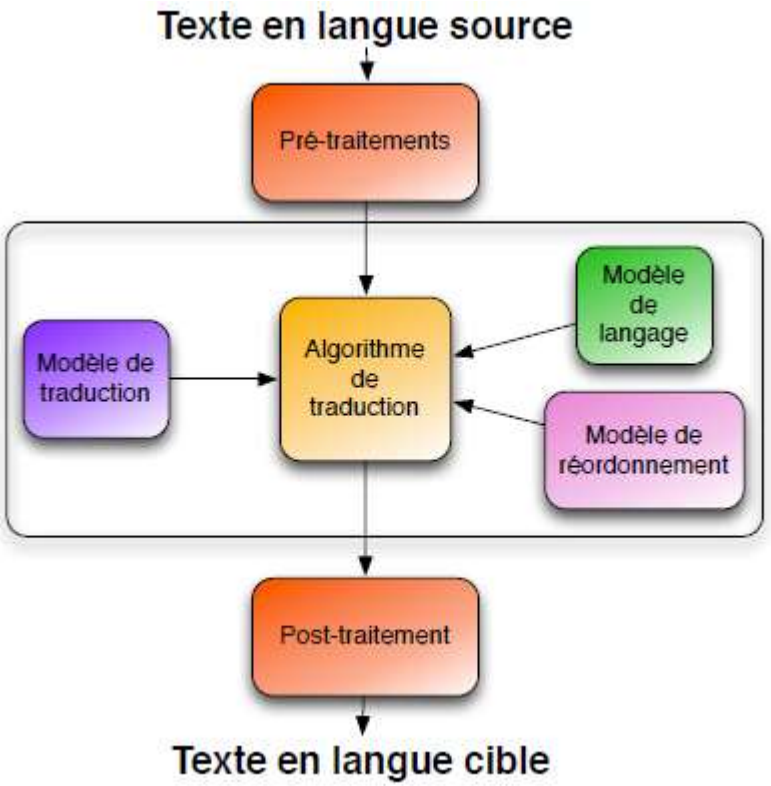
Définition: argmax est l'ensemble des points en lesquels une fonction atteint sa valeur maximale

$$\textbf{argmax}\ f = \{x \in X \mid \forall x' \in X,\ f(x') \leq f(x)\}$$

Système de Traduction Statistique



Architecture d'un Système de Traduction Statistique



Modèle de Traduction

- Approche statistique :

$$t* = \underset{t}{\operatorname{argmax}} P(s|t)P(t)$$

- Comment estimer P(s|t) pour toutes les séquences de mot *t* ?
  - Fréquences relatives : $p(s|t) = \frac{C(s,t)}{C(t)}$
  - Impossible !
    - s et t sont des phrases trop longues, et donc peu fréquentes
    - l'estimation des probabilités n'est pas bonne
    - ➡ il faut décomposer !

Modèle de Traduction: Décomposition de P(s|t)

- Les phrases *s* et *t* sont décomposées en groupes de un ou plusieurs mots
- Chaque groupe est traduit séparément
- Petit à petit la phrase source est composée
- Des réordonnements de groupes sont possibles
- La probabilité de traduction est la somme sur tous les alignements possibles :

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

Modèle de Traduction: Alignement de mots

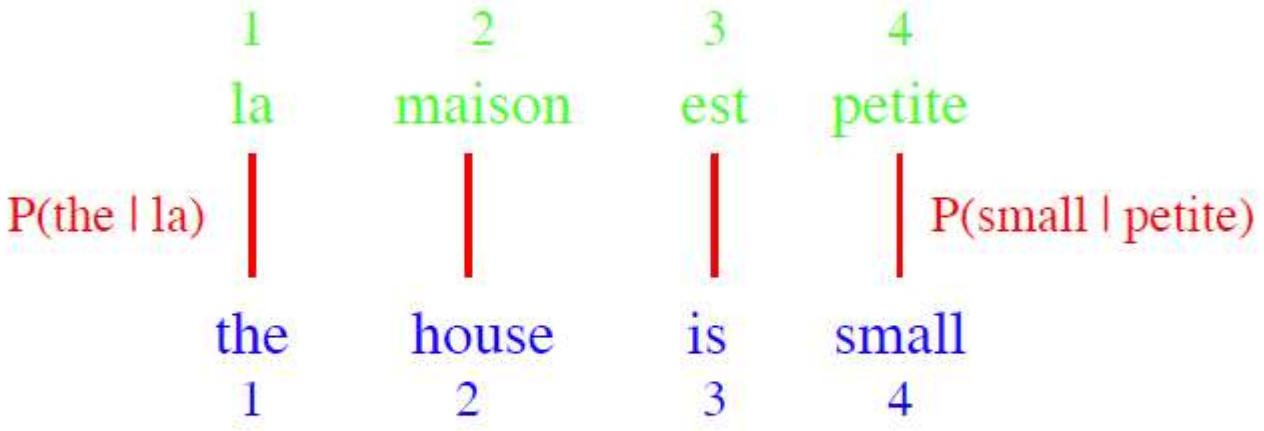
- L'unité de traduction est le mot
- Le modèle de traduction devient alors

$$p(a, s|t) = \prod_{j=1}^{m} lex(s_j|t_i)$$

  - avec $lex(s_j|t_i)$ la probabilité que le mot s_j se traduise en t_i, dite **probabilité lexicale**

- Il faudrait un dictionnaire probabiliste !
  - contenant toutes les traductions d'un mot
  - avec leurs probabilités

Alignement de mots: Exemple



- Alignement ici : 1→1, 2→2, 3→3, 4→4

Alignement de mots: Calcul des probabilités lexicales

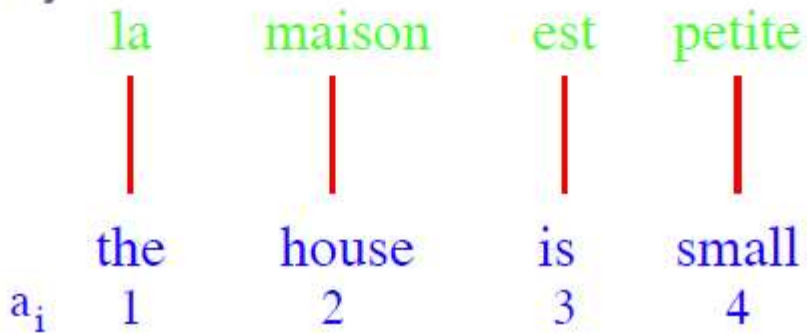
- Méthode : examiner un corpus
  - Exemple : mot source [fr] maison

Traduction	# d'observations	probabilité
house	670	0.67
building	60	0.06
home	254	0.254
household	15	0.015
shell	1	0.001

  - Probabilités déterminées par fréquences relatives
  - Nécessite un alignement des mots

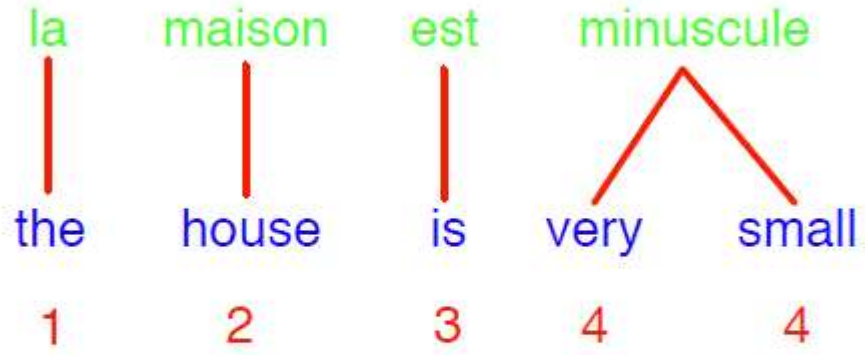
Alignement de mots: Formulation

- $s = s_1^m = s_1..s_j..s_m$
- $t = t_1^n = t_1..t_i..t_n$
- $a_1^n = a_1..a_n$ avec $a_i = j$, $j=1..n$, $i=1..m$;
- Le mot cible à la position i est connecté au mot source en position j



Alignement basé sur le mot

- Alignement 1 vers N



- 1 mot de la langue source se traduit en plusieurs mots de la langue cible

Alignement basé sur le mot

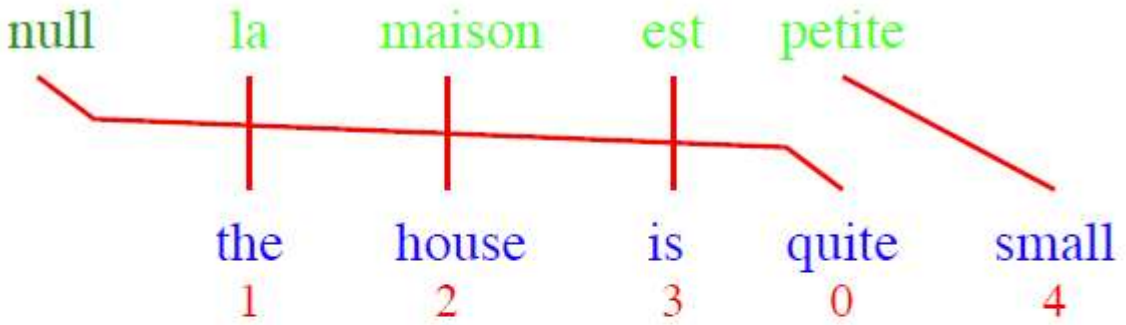
- Alignement N vers 1

- Impossible d'exprimer cela avec la notation mathématique

➡ Le mot source n'est aligné à aucun mot cible

- Autres exemples
  - [fr] ne ... pas → [en] not
  - [en] will learn → [fr] apprendra

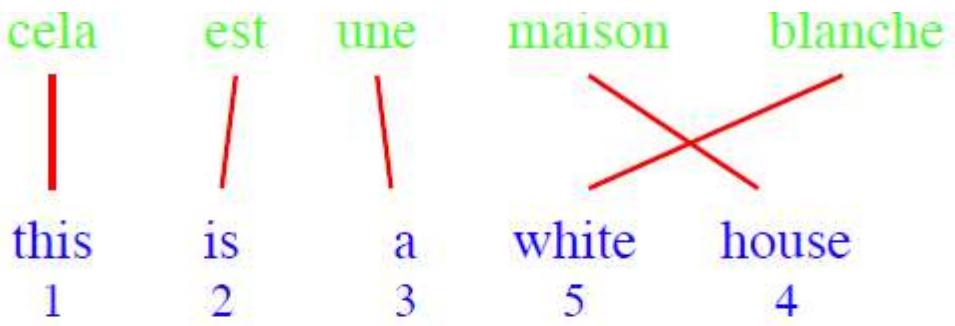
Alignement avec NULL



- Des mots en langue cible peuvent ne pas avoir de correspondance en langue source

➡ Ajout du mot spécial *null*

Alignement avec réordonnement



- Lien croisés = réordonnement
- Exemples typiques
  - [fr] substantif adjectif → [en] adjectif substantif
  - verbe adverbe → adverbe verbe

Modèles d'alignement: IBM 1

- Traductions lexicales (basées sur les mots)
- Données
  - 1 phrase source S = (s_1, …, s_m) de taille m
  - 1 phrase cible T = (t_1, …, t_n) de taille n
  - avec une fonction d'alignement a : j → i

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

  - ϵ est une constante de normalisation

Modèles d'alignement: IBM 1 - Exemple

la

c	lex($c\|s$)
the	0.75
that	0.15
which	0.045
who	0.03
this	0.025

maison

c	lex($c\|s$)
house	0.8
building	0.16
home	0.02
household	0.015
shell	0.005

est

c	lex($c\|s$)
is	0.7
's	0.26
exists	0.025
has	0.01
are	0.005

petite

c	lex($c\|s$)
small	0.4
little	0.4
short	0.1
minor	0.06
pretty	0.04

$$
\begin{aligned}
p(T,a|S) &= \frac{\epsilon}{4^3} \cdot lex(the|la) \cdot lex(house|maison) \cdot lex(is|est) \cdot lex(small|petite) \\
&= \frac{\epsilon}{5^4} \cdot 0.75 \cdot 0.8 \cdot 0.7 \cdot 0.4 \\
&= 0.001344\epsilon
\end{aligned}
$$

Modèles d'alignement: IBM 1 – Apprentissage des probabilités lexicales

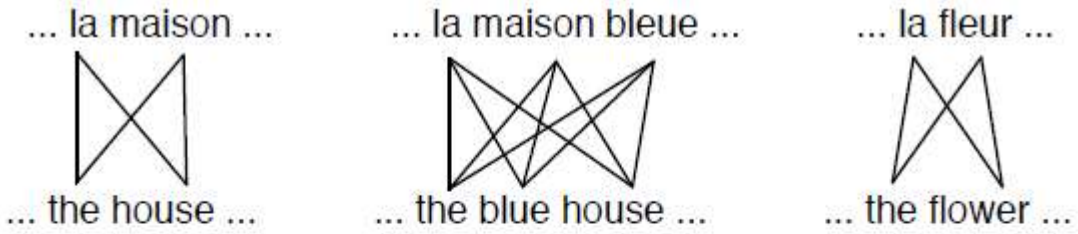
- à partir d'un corpus parallèle
  - il manque les alignements !
- Problème
  - Si on avait les alignements, on pourrait estimer les paramètres du modèle
  - Si on avait les paramètres, on pourrait estimer les alignements

Apprentissage des probabilités lexicales: Algorithme EM

- Expectation / Maximization
  - Données incomplètes → manque l'alignement
- Principe d'EM :
  - initialiser les paramètres du modèle (uniformément)
  - affecter des probabilités aux données manquantes
    - Expectation
  - estimer les paramètres avec les données complètes
    - Maximization
  - réitérer étapes 2 et 3 jusqu'à convergence

Définition: L'algorithme EM (Expectation-Maximization - Espérance-Maximisation est un algorithme itératif qui permet de trouver les paramètres du maximum de vraisemblance d'un modèle probabiliste.

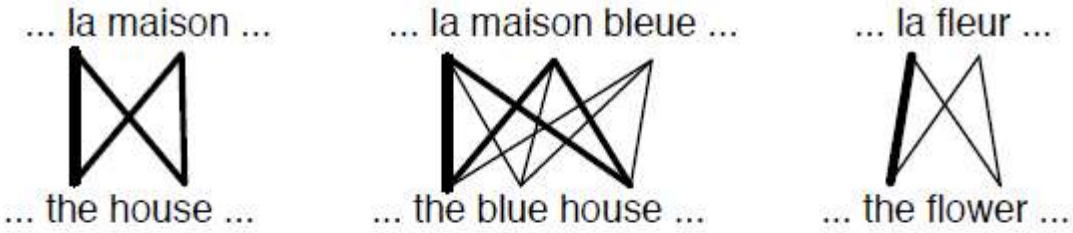
Apprentissage des probabilités lexicales: Algorithme EM



- Étape initiale : tous les alignements sont équiprobables
- Le modèle apprend
  - **la** est souvent aligné avec **the**
  - **maison** avec **house**

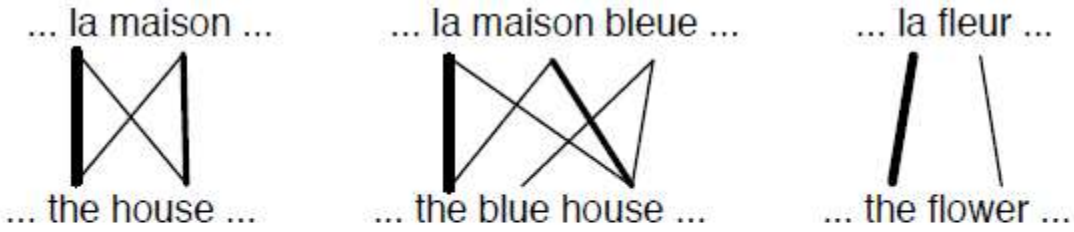
  → besoin de grands corpus bilingues (bitextes)

Apprentissage des probabilités lexicales: Algorithme EM



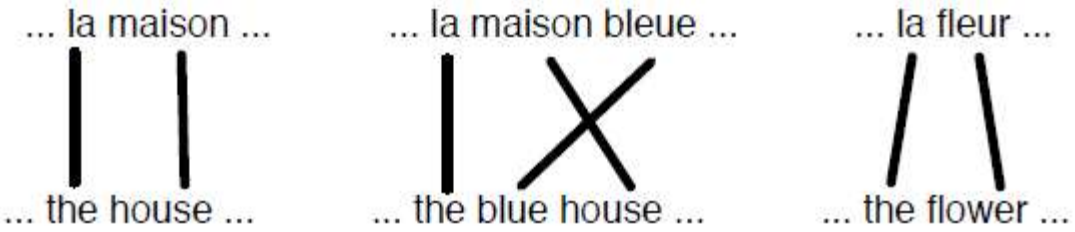
- Après 1 itération
- Alignements
  - entre **la** et **the** sont + probables
  - idem **maison** et **house**
  - mais aussi **la** et **house**, et **maison** et **the**

Apprentissage des probabilités lexicales: Algorithme EM



- Après une autre itération
- les alignements entre **fleur** et **flower** sont + probables

Apprentissage des probabilités lexicales: Algorithme EM



- Après convergence
- La structure cachée est révélée par EM

→ On peut calculer les paramètres du modèle
  - p(la|the), p(maison|house), etc …

Modèle IMB 1 et Algorithme EM

- Expectation : appliquer le modèle aux données
  - assigner des probabilités aux alignements
  - en fonction des connaissances actuelles
- Maximization : estimer le modèle à partir des données
  - collecter les comptes (pondérés par les probabilités)
  - estimer les paramètres à partir de ces comptes
- et on recommence jusqu'à convergence

Algorithme EM - Expectation

- Nous voulons calculer $p(a|S,T)$
- Règle de Bayes $p(a|S,T) = \frac{p(T,a|S)}{p(T|S)}$
- On connait déjà le numérateur (modèle IBM1)

Théorème de Bayes: Déterminer la probabilité qu'un événement arrive à partir d'un autre évènement qui s'est réalisé, notamment quand ces deux évènements sont interdépendants.

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

Algorithme EM - Expectation

- Il faut calculer $p(T|S)$

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

- Note : l'astuce en dernière ligne permet de rendre cela calculable (supprime le nombre exponentiel de produits)

Algorithme EM - Expectation

- En combinant le tout, on obtient :

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

Algorithme EM - Maximization

- Il faut collecter les comptes

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

- avec la même simplification que précédemment :

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

Algorithme EM - Maximization

- On peut maintenant estimer le modèle :

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

Perplexité

- À quel point le modèle correspond aux données ?
- Perplexité : mesure calculée à partir de probabilités des données d'entraînement en fonction du modèle

$$\log_2 PP = -\sum_i \log_2 p(t_i|s_i)$$

	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

Alignement d'expressions

- Cela peut se compliquer rapidement

- ➡ Limitations de l'alignement mot à mot
- ➡ Il faudrait un alignement de groupes de mots
  - Simplification
  - Plus réaliste

Alignement par groupe de mots

- Lorsque l'on connait l'alignement, on peut calculer :

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

- On dispose de textes parallèles alignés au niveau de la phrase (Europarl, U.N., etc.)
- Mais : très couteux et difficile de faire un alignement manuel

➡ Cet alignement doit être déterminé automatiquement

Alignement dans les deux directions

- Effectuer les alignements mot à mot dans les 2 sens

- Rappel : l'alignement est **asymétrique** !

Alignement dans les deux directions

- Effectuer les alignements mot à mot dans les 2 sens

- Rappel : l'alignement est **asymétrique** !

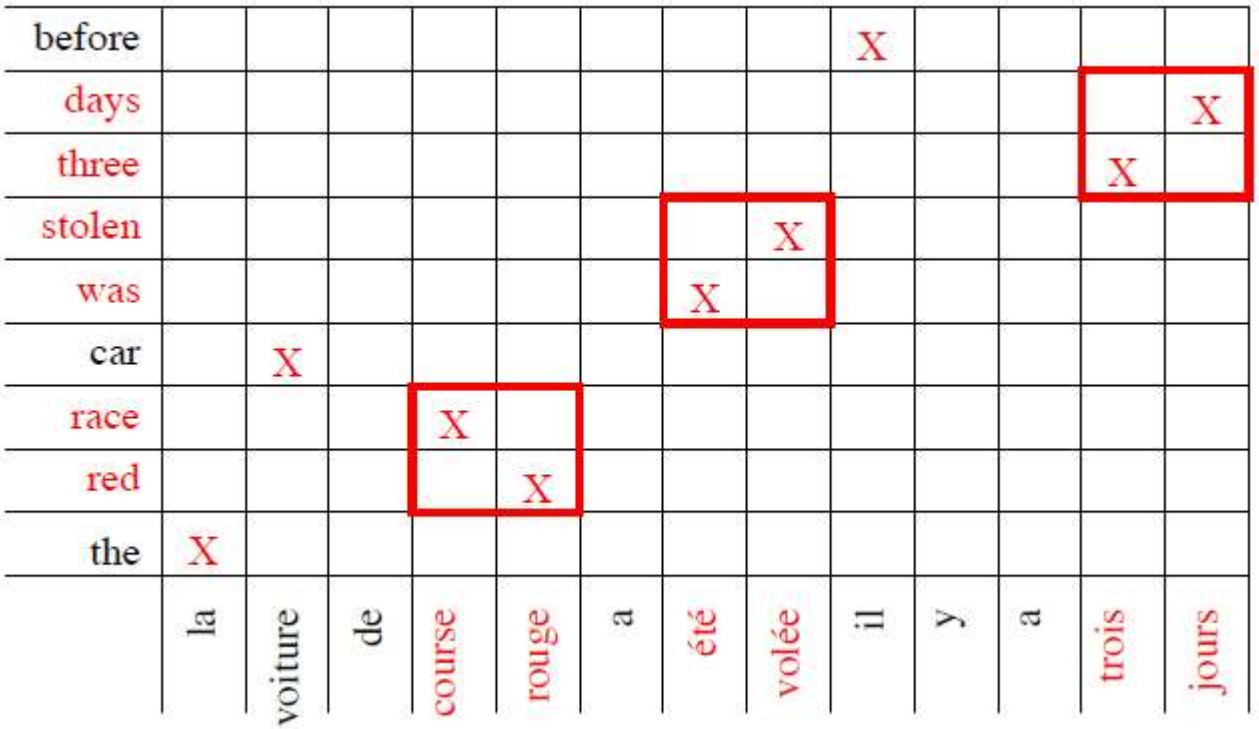
Etape 1: Matrice d'alignement

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

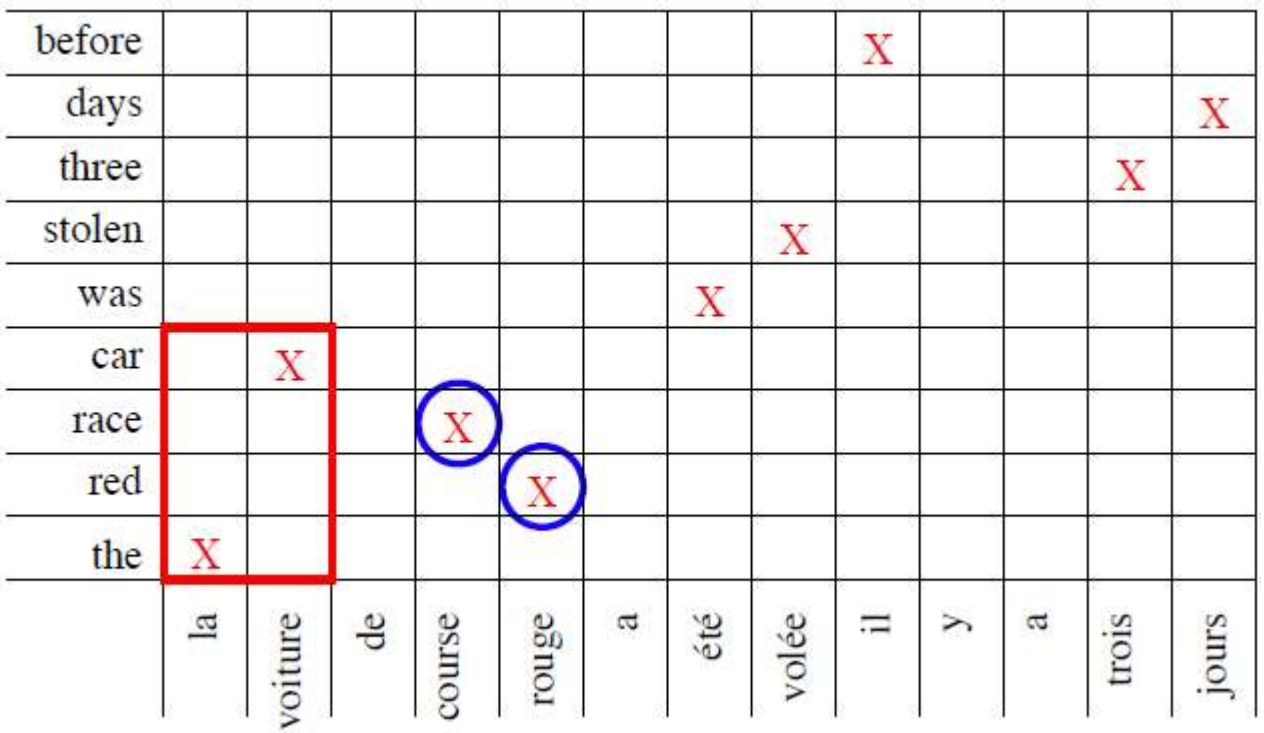
Etape 2: Marquer l'intersection des alignements

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

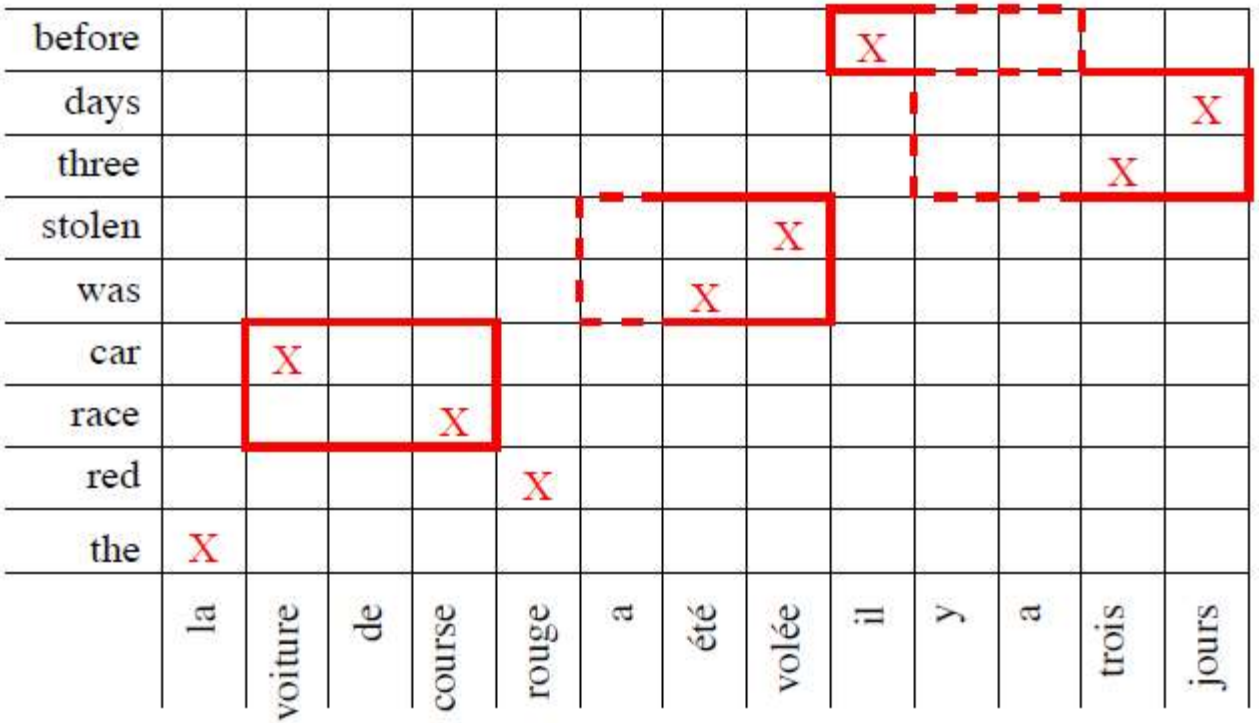
Etape 3: Marquer les blocs de mots



Etape 3: Marquer les blocs de mots - Exemple de bloc non autorisé



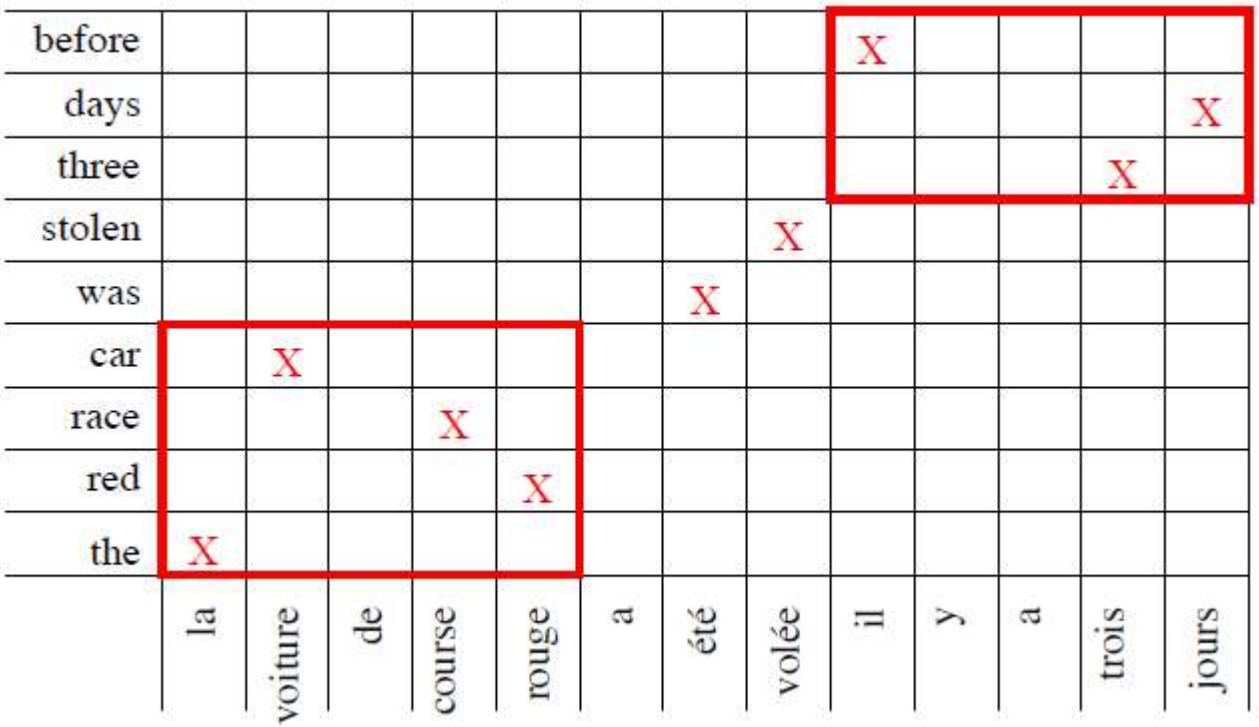
Etape 3: Marquer les blocs de mots - Extension des blocs de mots



Etape 3: Marquer les blocs de mots - blocs de maximum 4 mots

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

Etape 3: Marquer les blocs de mots - blocs de maximum 5 mots



Construction de la table de traduction

- Utilisation des fréquences relatives

Source s	Cible t	#occurences	P(t|s)	P(s|t)
I go	je vais	123	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
...	...	...	...	...

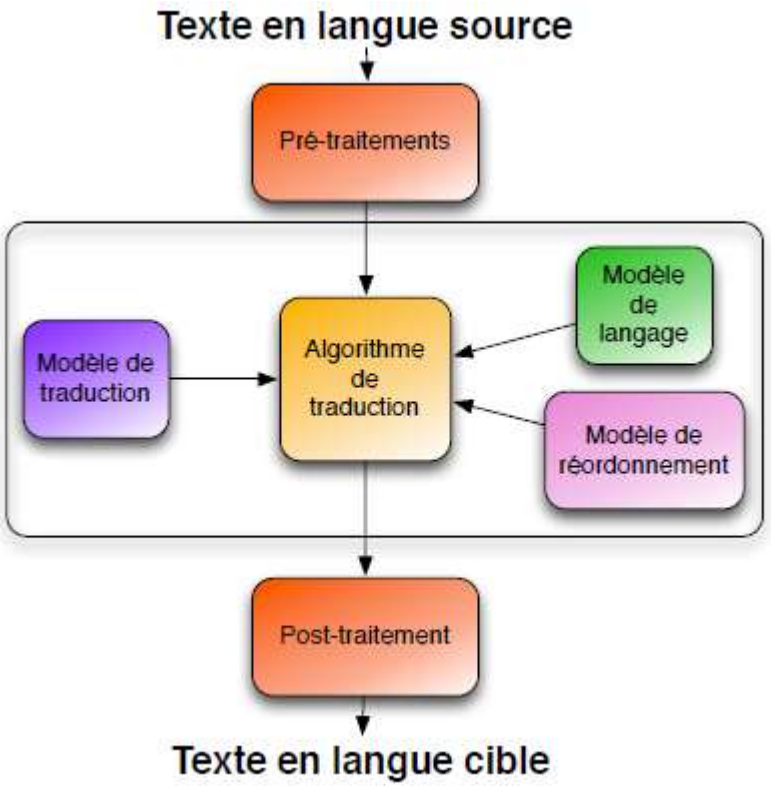
- Utilisation des probabilités inverses
  - Différentes car alignements asymétriques !

Scores de la table de traduction

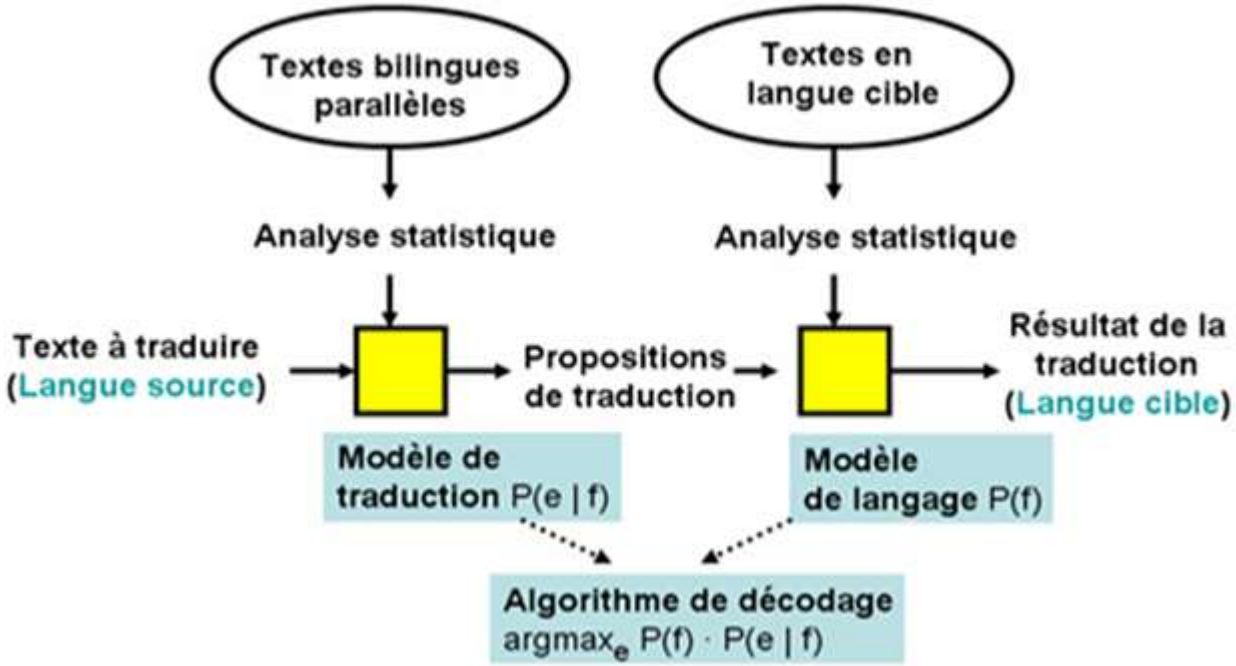
- 5 scores sont présents
  - p(s|t)
  - lex(s|t)
  - p(t|s)
  - lex(t|s)
  - pénalité d'insertion de phrase-pair = e^1 : 2.718
- avec

$$lex(\tilde{s}|\tilde{t}) = \prod_{i,j} lex(s_j|t_i)$$

Architecture d'un Système de Traduction Statistique



Modèle de langage



Modèle de langage

- Composante importante du système
- Permet de faire ressortir les phrases “bien formées” des autres
- Doit privilégier les phrases grammaticalement et sémantiquement correctes
  - de manière implicite
  - aucune analyse grammaticale ni sémantique

Modèle de langage

- Objectif : associer une probabilité non-nulle à **toutes** les suites de mots
  - doit permettre de traiter les phrases agrammaticales
  - apprentissage automatique à partir de textes
- Problèmes :
  - Comment associer une probabilité aux séquences non-observées ?
  - Limitation du contexte
    - taille du modèle
    - estimation des probabilités

Modèle de langage: Rôle

- Assigner une probabilité non-nulle à toutes les séquences de mots W extraites du vocabulaire V
- Vocabulaire = liste des mots connus par le modèle
  - + une classe pour les mots inconnus <unk>
  - mot = séquence de caractères sans espace
  - différent d'un mot en linguistique (token)

$$W = (w_1, w2, \ldots, w_n) \quad \text{avec} \quad w_i \in V$$

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

$$\text{avec} \quad h_i = (w_1, w_2, \ldots, w_{i-1}) \quad \text{l'historique de} \quad w_i$$

Modèle de langage: Complexité

- Complexité (vocabulaire 65k)
  - 65000^2 = 4 225 000 000 suites des 2 mots possibles
  - 65000^3 = 2,74 x 10^{14} suites des 3mots possibles
- Classe d'équivalences
  - Regrouper les historiques en classes d'équivalence ϕ

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

  - Citation F. Jelinek :
  - Tout l'art de la modélisation de la langue consiste à déterminer ϕ

Modèle de langage: n-grams

- La probabilité ne dépend que des n-1 mots précédents
- bigram : $\phi(h_i) = (w_{i-1})$

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

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

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

- n-gram : $\phi(h_i) = (w_{i-1}, \ldots, w_{i-n+1})$
- Conséquences
  - n-1 mots suffiraient à prédire le mot suivant
  - en pratique $n \leq 4$

Modèle de langage: Caractéristiques

- Structure de la langue capturée implicitement
  - probabilité de succession de mots
  - idem pour la sémantique
- Probabilités indépendantes de la position dans la phrase
  - corrigé par l'ajout d'indicateur de début et fin de phrases
  - <s> et </s>
- Probabilités estimées à partir de grands corpus de textes **supposés bien écrits**

Modèle de langage: Calcul des probabilités

- Probabilités des unigrammes

$$p(w_i) = \frac{C(w_i)}{\sum_k C(w_k)} = \frac{C(w_i)}{taille\ du\ corpus}$$

- Probabilités n-gram

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

Modèle de langage: Événements non observés

- Séquences non admises par la langue
  - ex: “ils part tôt”
- Séquences absentes du corpus
- Solutions :
  - augmenter la taille du corpus d’apprentissage
  - Attribuer une probabilité (faible) aux événements non observés :

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

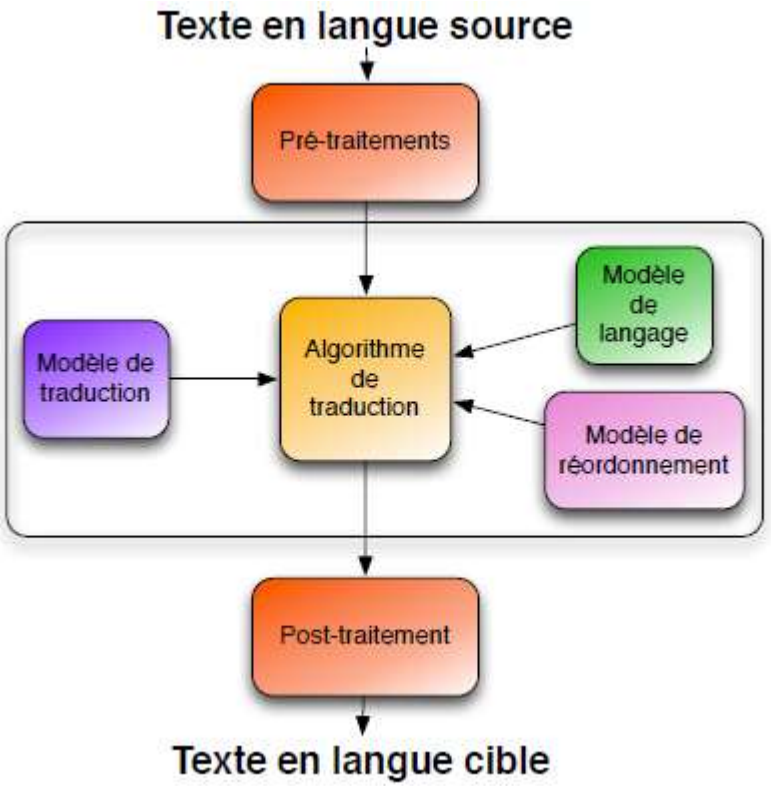
Qualité d'un Modèle de Langue – Perplexité

- Correspond à la capacité du ML à prédire les mots d'un texte inconnu
  - inconnu = non inclus dans les données d'entraînement
  - il faut minimiser ce critère (optimisation)

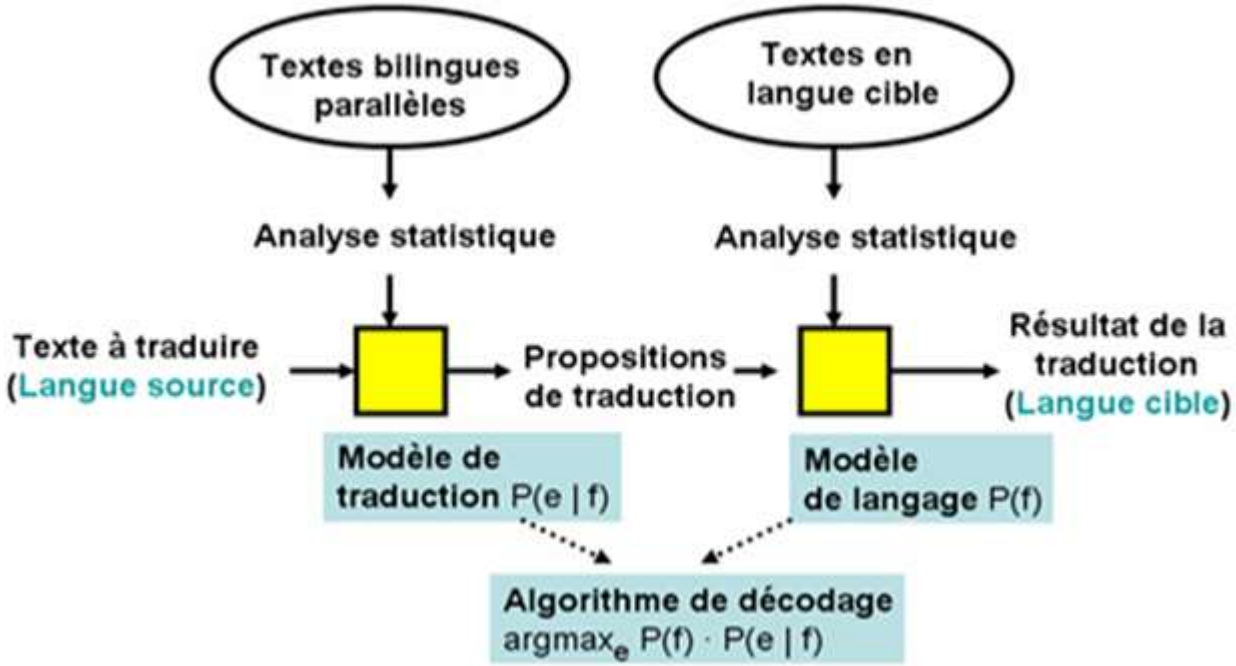
$$
\begin{aligned}
PPL &= 2^{H} \\
&= \log \prod_{i=1}^{T} p(w_i)^{-\frac{1}{T}} \\
&= \sum_{i=1}^{T} \log p(w_i)^{-\frac{1}{T}} \\
&= -\frac{1}{T} \sum_{i=1}^{T} \log p(w_i)
\end{aligned}
$$

- Somme des log
- Normalisation par T

Architecture d'un Système de Traduction Statistique



Décodeur


Textes bilingues parallèles
Textes en langue cible
Analyse statistique
Analyse statistique
Texte à traduire (Langue source)
Propositions de traduction
Résultat de la traduction (Langue cible)
Modèle de traduction P(e | f)
Modèle de langage P(f)
Algorithme de décodage
argmax$_e$ P(f) · P(e | f)

Décodage

- Rappel du modèle statistique :

$$t* \quad = \quad \underset{t}{\operatorname{argmax}}\, P(s|t)P(t)$$

- Nous savons calculer *P(s|t)* et *P(t)*
- Il faut trouver *t** : c'est le rôle du décodeur

Décodage: Principe

- parcours de la phrase source
- création d'hypothèses de traduction
  - itération 1 : 1 mot source couvert
  - itération 2 : 2 mots source couverts
  - etc ...
- Ex : la voiture de course rouge a été volée il y a trois jours

Décodage: Complexité

- Décodage : problème NP-complet
- Ce processus génère un nombre exponentiel d'hypothèses
- Solution : il faut réduire l'espace de recherche !
  - recombinaison des hypothèses partielles - sans pertes
  - *pruning* ou élagage - avec pertes

Remarque:
La plupart des algorithmes connus pour résoudre des problèmes NP-complets ont un temps d'exécution exponentiel en fonction de la taille des données d'entrée, et sont donc inexploitables en pratique même pour des instances de taille modérée.

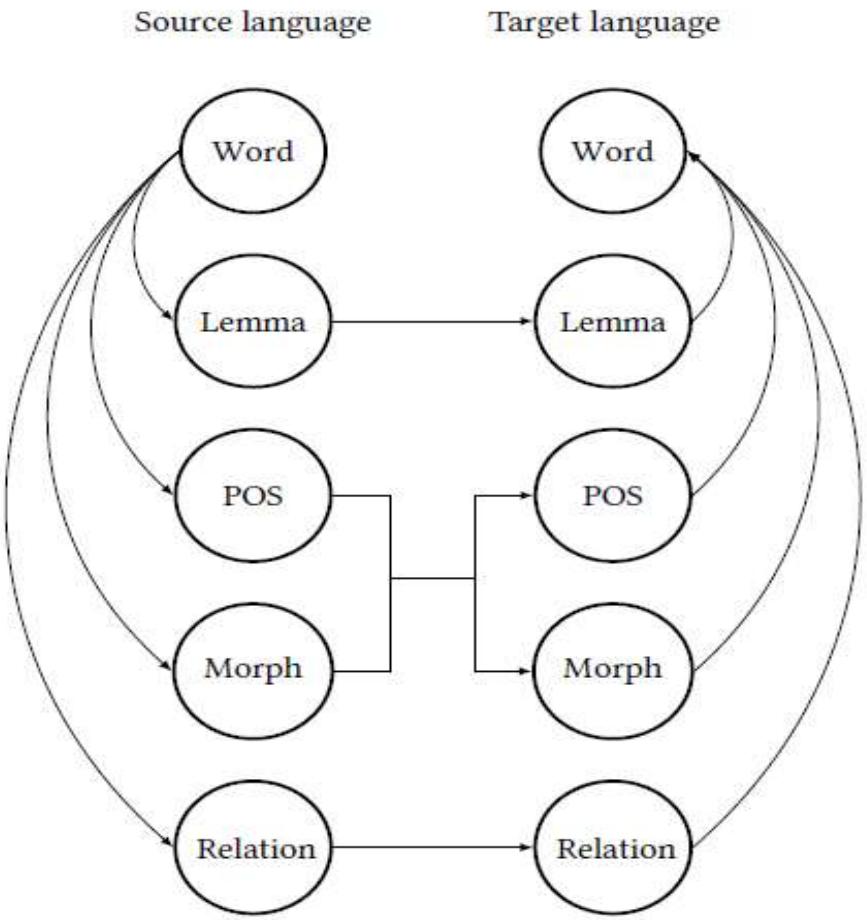
Décodage: Modèle Log-linéaire pondéré

- En réalité

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

- h : fonction paramétrique
  - p(s|t), p(t|s), lex(s|t), lex(t|s), plm(t), etc …
- Certains paramètres peuvent être plus importants que d'autres
  - ajout d'un poids λ_i
  - ces poids sont optimisés
  - ex: MERT : Minimum Error Rate Training

Traduction statistique: Factored model



The model operates on lemmas instead of surface forms, in which the translation process is broken up into a sequence of mapping steps:

1. *Translate source lemmas into target's ones.*
2. *Generate surface forms given the lemma.*

Traduction statistique: Conclusion

- Avantages
  - Les connaissances sont extraites automatiquement
  - Intervention humaine minimale
- Inconvénients
  - La syntaxe n'est pas gérée explicitement par le modèle de base
  - Pas de mécanisme de généralisation
    - On peut savoir traduire «10 pommes rouges» mais ne pas savoir traduire «11 pommes rouges»
    - La connaissance se limite à ce qui est rencontré dans les bitextes
  - Difficile de prédire le résultat après une modification du système

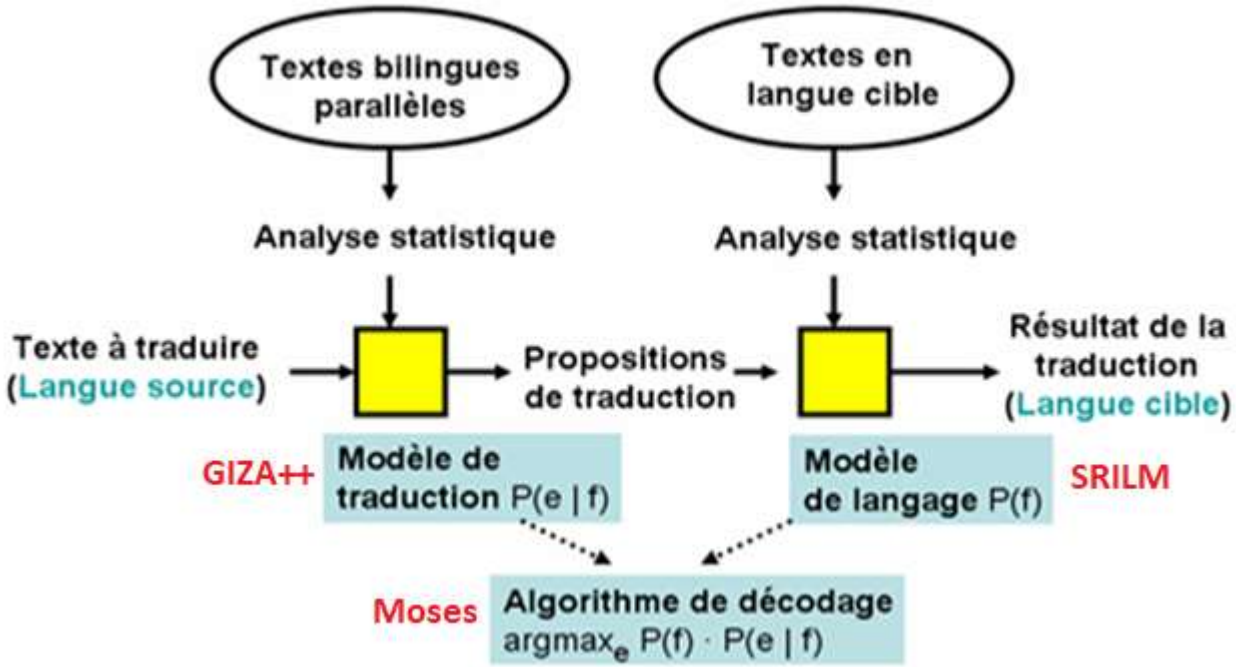
Traduction statistique: Conclusion

- Développement d'un système statistique
  - Rapide
  - Peu coûteux
  - Possibilité de traduire beaucoup de paires de langues
  - Ne nécessite pas d'outils TAL ni de ressources linguistiques spécifiques
- **Il suffit d'avoir un corpus parallèle !**

Traduction statistique: Logiciels open source disponibles

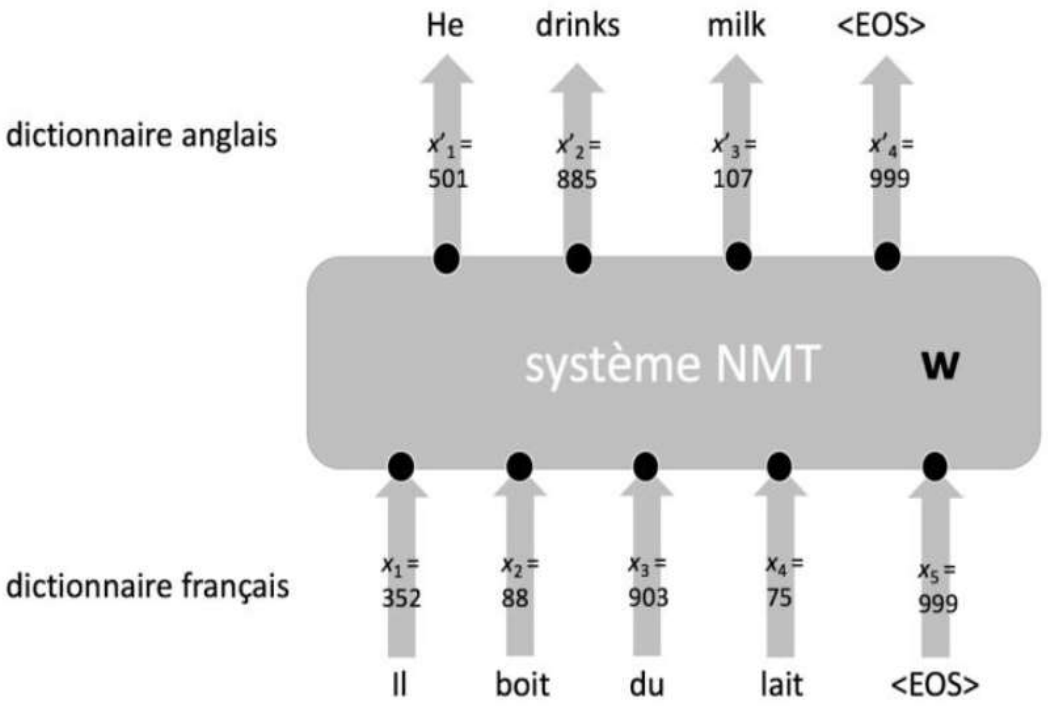
- **Moses** : http://statmt.org/moses
  - Boîte à outils : extraction de *phrase-pairs*, décodeur,
- **MERT/MIRA/PRO** : optimisation des poids
- Apprentissage des alignements
  - **GIZA++** : http://code.google.com/p/giza-pp
- Modèle de langue n-gram
  - **SRILM** : http://www.speech.sri.com/projects/srilm
  - **KenLM** : inclus dans Moses
- Tous ces outils sont disponibles, performants et permettent de construire des systèmes à l'état de l'art.

Traduction Statistique avec le toolkit Moses



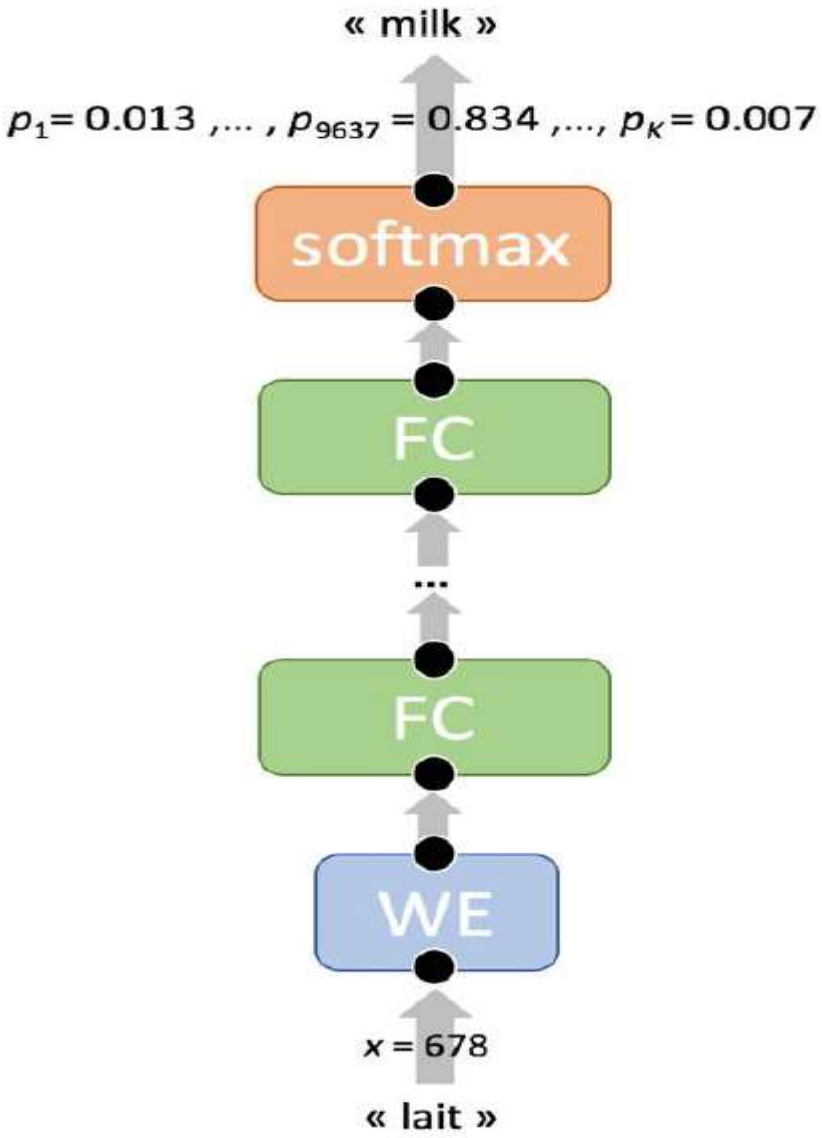
Traduction Neuronale

Système de traduction neuronale



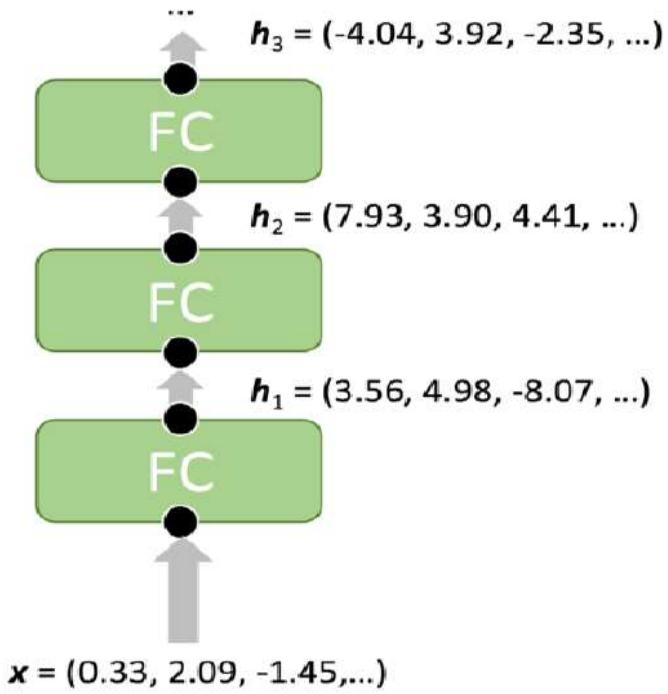
Un système de traduction automatique convertit une suite de token qui représente une phrase source en une autre suite de token qui représente la phrase traduite. Les paramètres **w** du système sont appris par ML supervisé.

Principaux composants d'un modèle de traduction neuronale



Fully-Connected layer

- Une couche FC (complètement connectée) désigne un réseau de neurones complètement connectés qui combine une transformation linéaire et une non-linéarité définie par une fonction d'activation
  - ➔ Définit une fonction (avec des paramètres ajustables w) qui transforme un vecteur en entrée $x=(x_1, x_2, \ldots, x_n)$ en un autre vecteur $h=(h_1, h_2, \ldots, h_m)$ en sortie
  - ➔ Ce vecteur h pourra à son tour jouer le rôle de vecteur en entrée d'une seconde couche et ainsi de suite
  - ➔ Les vecteurs intermédiaires entre deux telles couches sont appelées des variables cachées

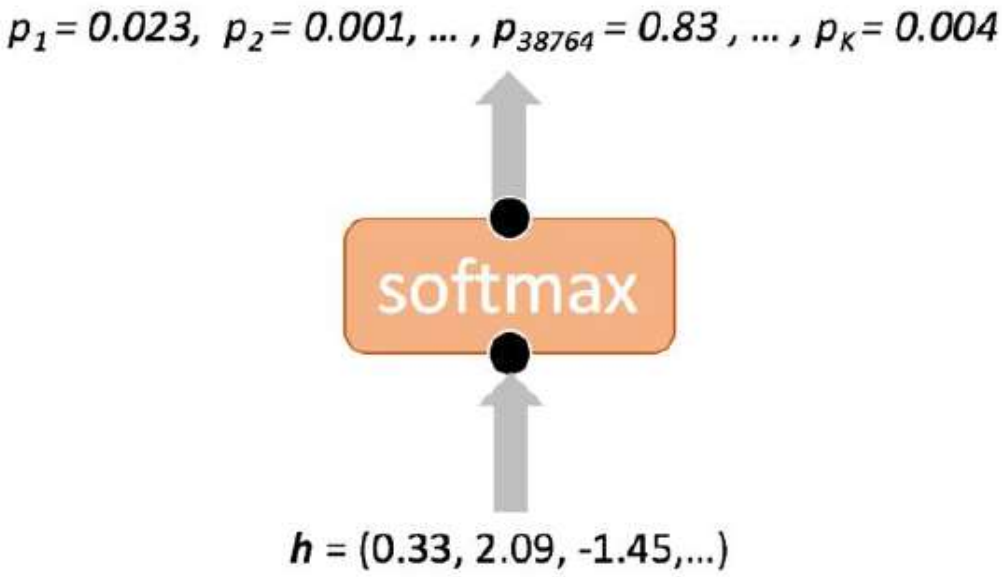


Couche Softmax

- Une couche Softmax convertit un vecteur en une distribution de probabilité sur les K tokens d'un vocabulaire

➔ Transforme le vecteur $\mathbf{h}_j$ en sortie du dernier module FC en une distribution de probabilité p_1, p_2, … p_K (dont la somme vaut 1) sur les entiers compris entre 1 et K

➔ Retient le token le plus probable



Entraînement du modèle de traduction neuronale mot à mot

- Présenter des mots en entrée (et par la suite des phrases) ainsi que leur traduction correcte

  ➔ Chaque traduction correcte d'un mot peut être envisagée comme une distribution qui attribue une probabilité 1 au mot correct et 0 à tous les autres (one-hot encoding)

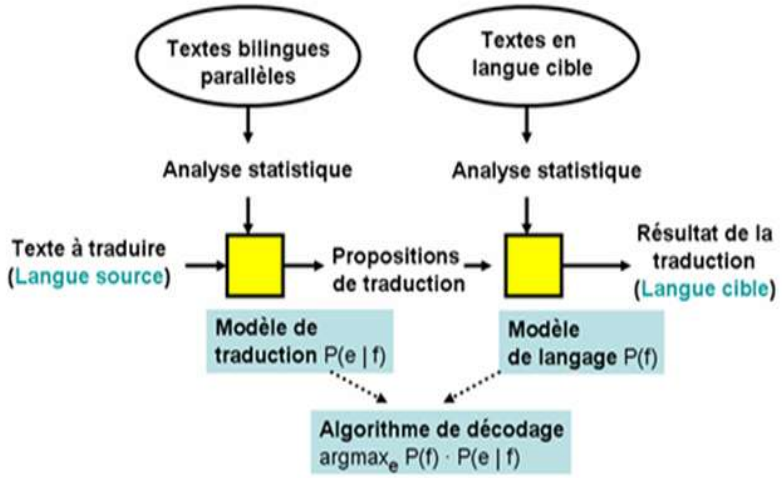
  ➔ Comparer, pour chaque mot du vocabulaire source, cette distribution correcte à celle prédite par le modèle de traduction neuronale

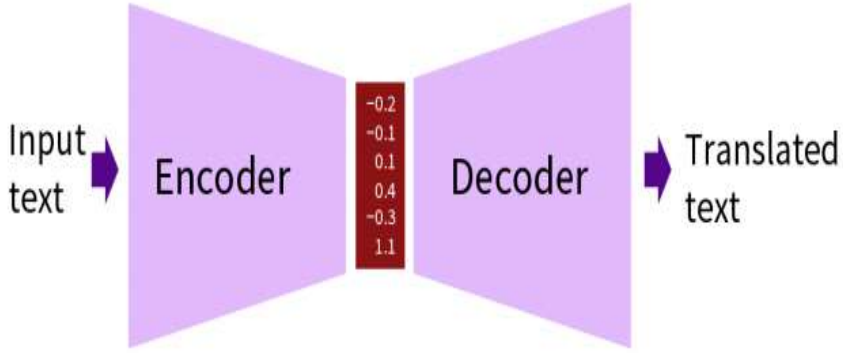
- Entraîner le modèle revient à chercher des paramètres **w** tels que, en moyenne sur tous les mots du vocabulaire source, ces deux distributions soient aussi proches que possible

  ➔ La technique d'entraînement utilisée est la rétro-propagation

  ➔ La cross-entropie est utilisée pour mesurer la proximité entre deux distributions de probabilité

SMT vs NMT (Architecture)





- Nombreux composants
=> Architecture en pipeline

- Modules indépendants (TM: Modèle de traduction, LM: Modèle de langue)

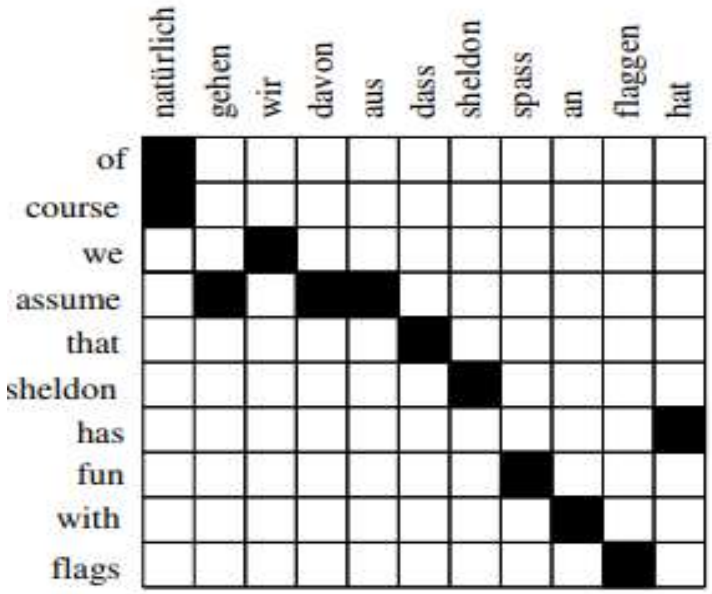
- Apprendre des pondérations en utilisant la formule de Bayes

- Un seul modèle

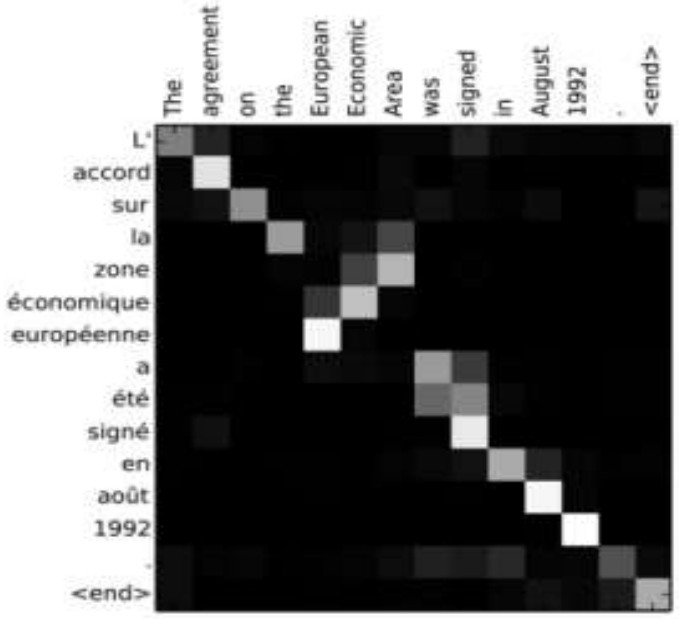
- Tous les paramètres sont optimisés

- Aucune division explicite en TM et LM

NMT et SMT (Alignement de mots)



Mécanisme d'alignement (correspondence entre les mots et leur traduction)



Modèle d'attention: Une représentation graphique des zones d'attention. Les carrés clairs sont ceux pour lesquels l'attention est importante

Le mécanisme d'attention apprend à traduire séquentiellement une suite de mots et simultanément à aligner les mots cibles avec les mots sources les plus pertinents

Les modèles NMT

- La traduction neuronale (NMT) n'est rien d'autre qu'une variante de la traduction statistique (SMT): P(e|f) est apprise en utilisant un réseau de neurones

- La traduction est modélisée comme un problème de prédiction et ressemble au modèle de langue mais la source et la cible sont toutes les deux apprises conjointement

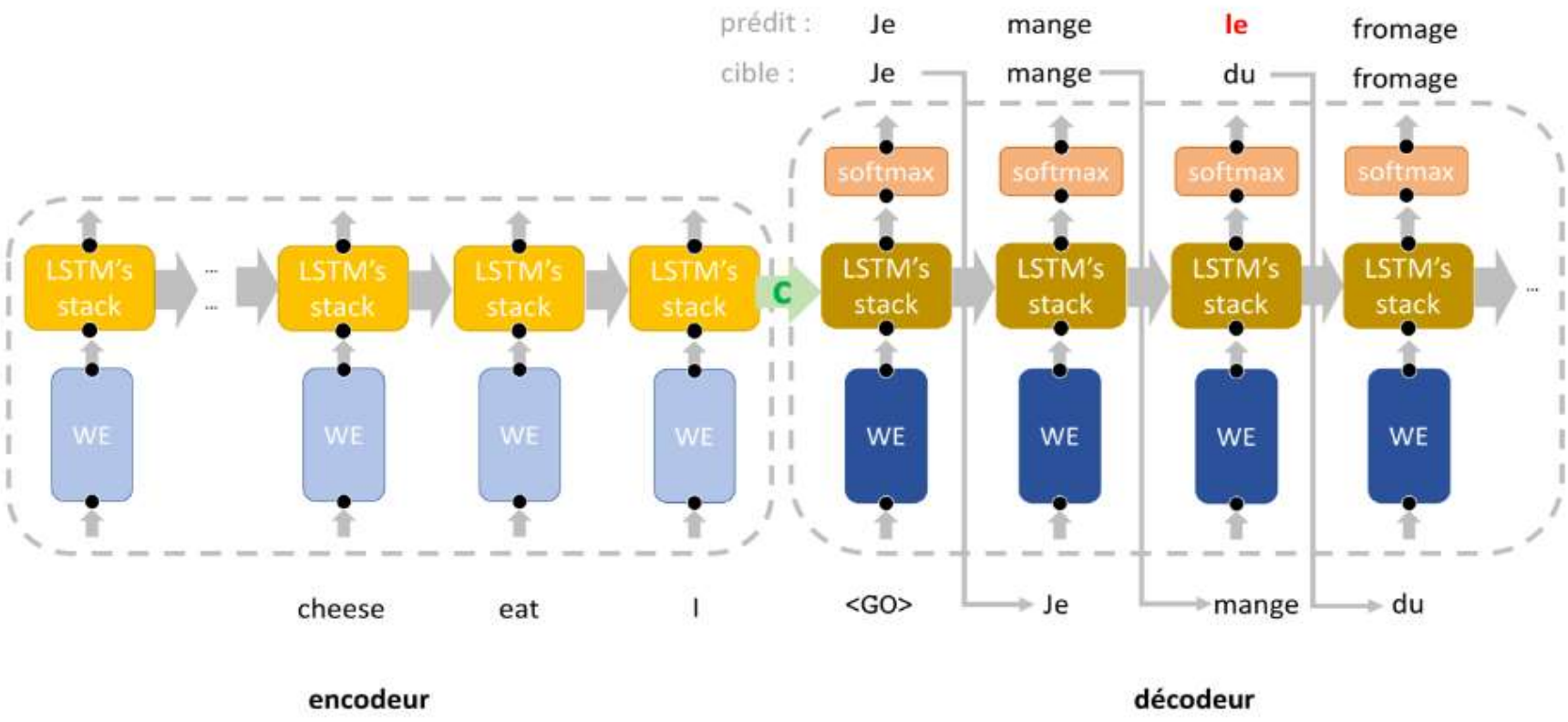
- Le modèle Encodeur – Décodeur utilise des RNNs ou des Transformers pour lire la source et prédire la cible

Les modèles NMT (Encodeur- Décodeur)

- **Encoder la phrase source:**
  - Convertir la phrase en un vecteur de taille fixe
  - Mapper le vecteur obtenu aux états cachés en utilisant des réseaux de neurones récurrents (ou Transformers)

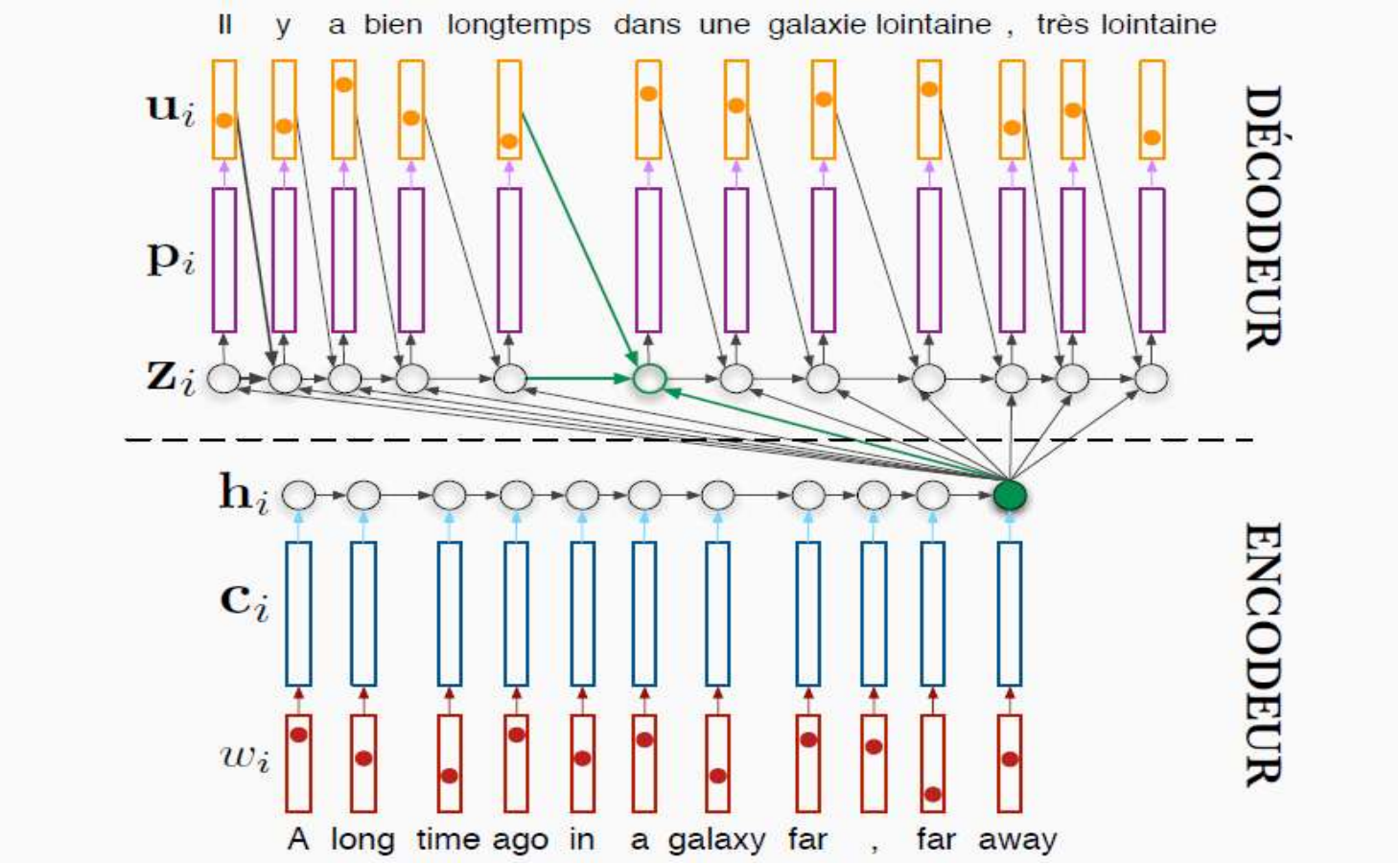
- **Décoder vers la phrase cible:**
  - Le premier état caché du décodeur est le dernier état caché de l'encodeur
  - Prédire les mots cibles en leurs affectant des probabilités
  - Choisir les mots avec une probabilité maximale
  - Mettre à jour l'état caché suivant avec la valeur prédite

Architecture des modèles NMT: Encodeur- Décodeur avec des LSTMs (sans mécanisme d'attention)

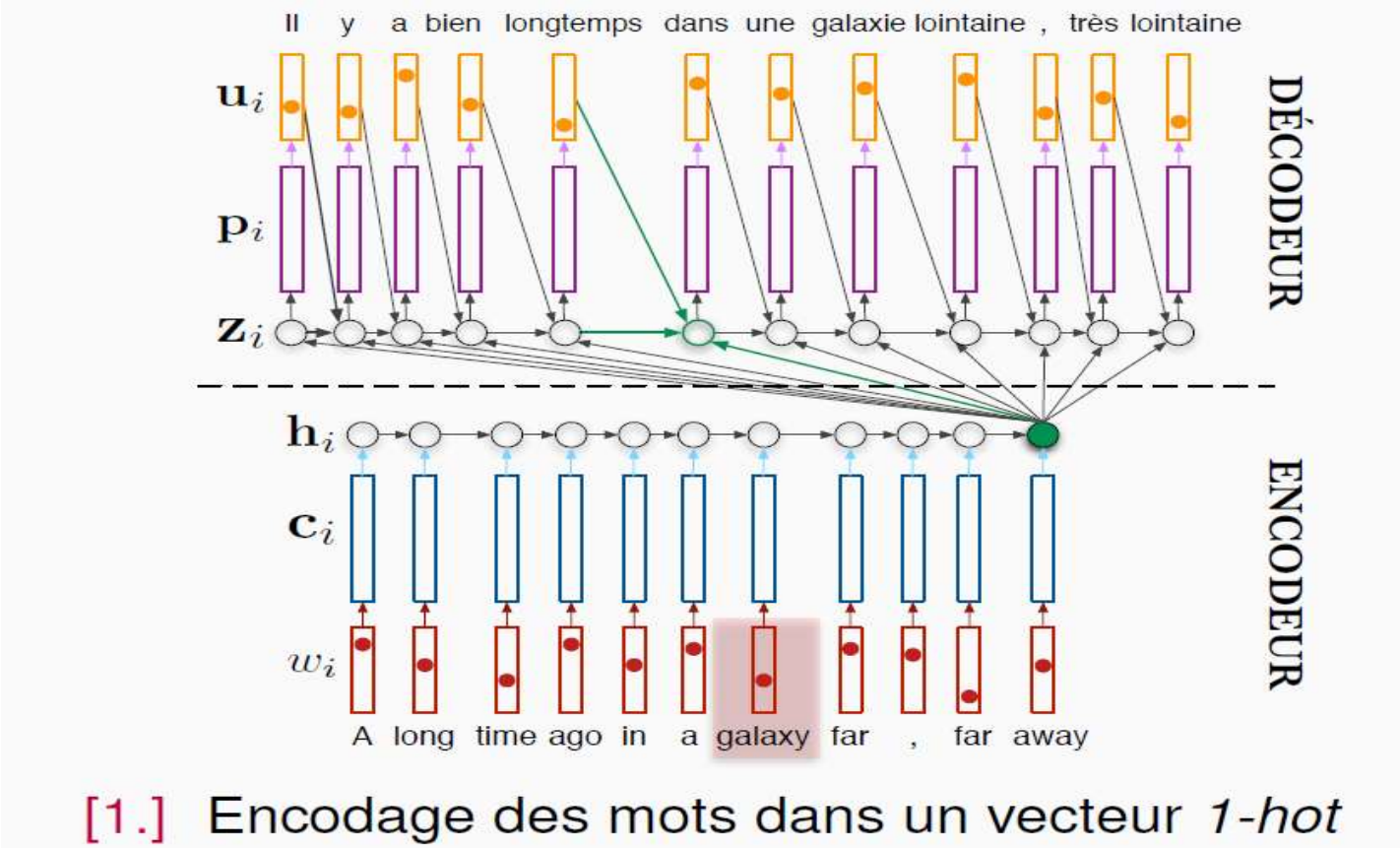


Le contenu du vecteur C (couche cachée) en bout de chaîne de l'encodeur contient le sens de la phrase

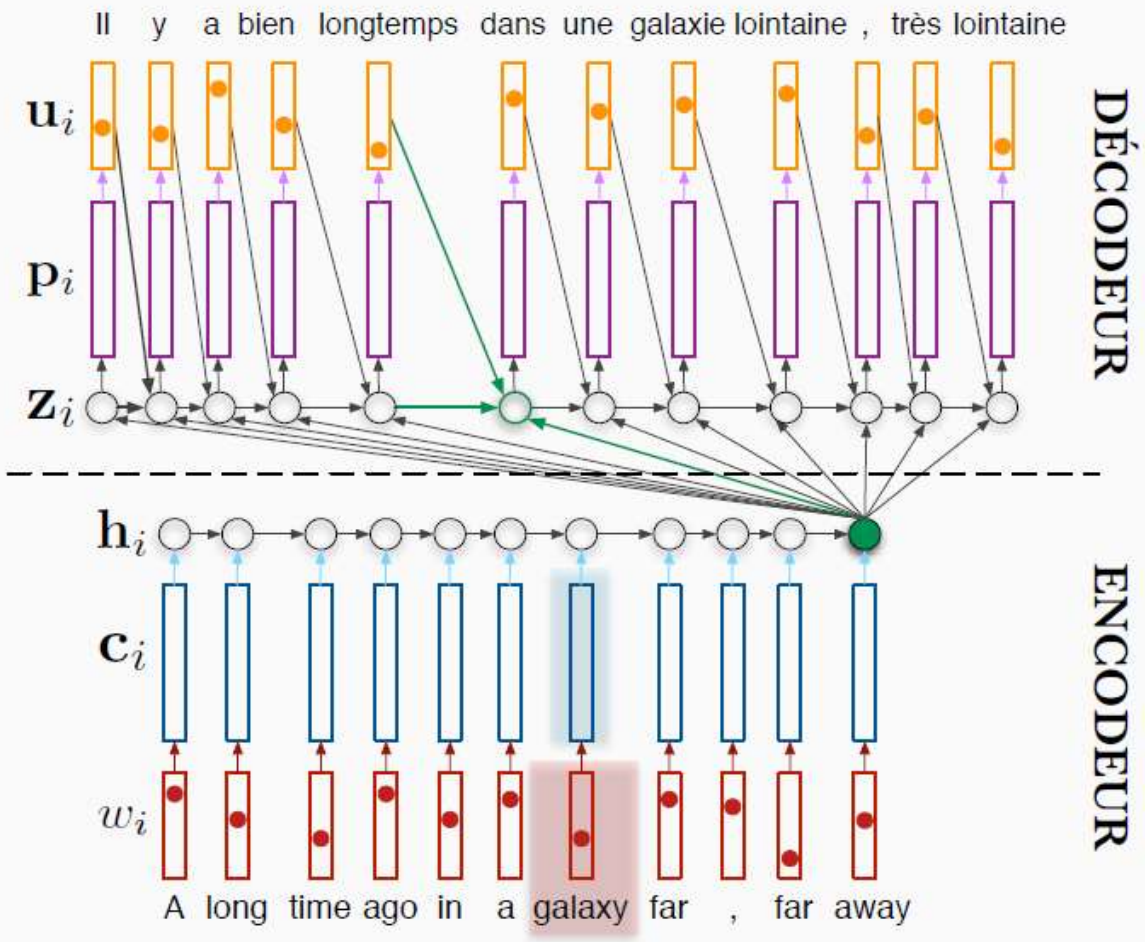
Architecture des modèles NMT: Encodeur- Décodeur (sans mécanisme d'attention)



Architecture des modèles NMT: Encodeur- Décodeur (sans mécanisme d'attention)

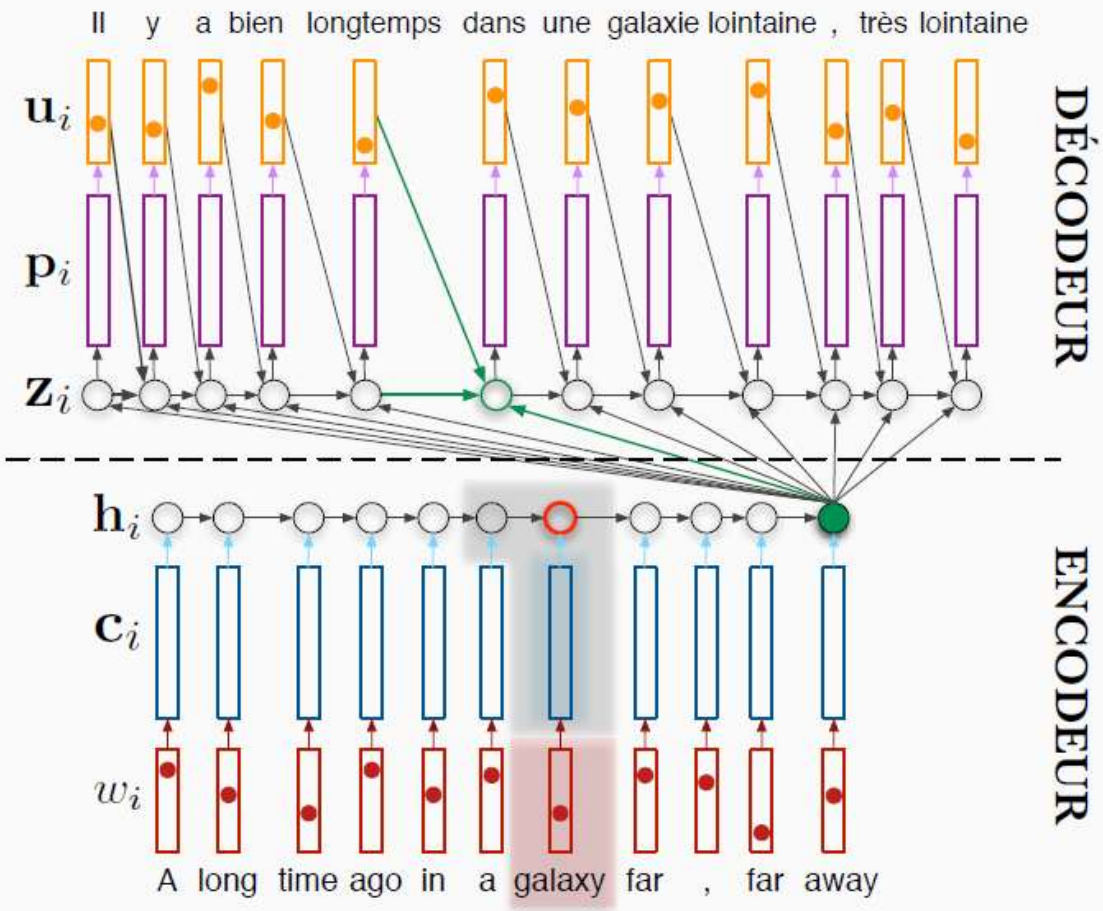


Architecture des modèles NMT: Encodeur- Décodeur



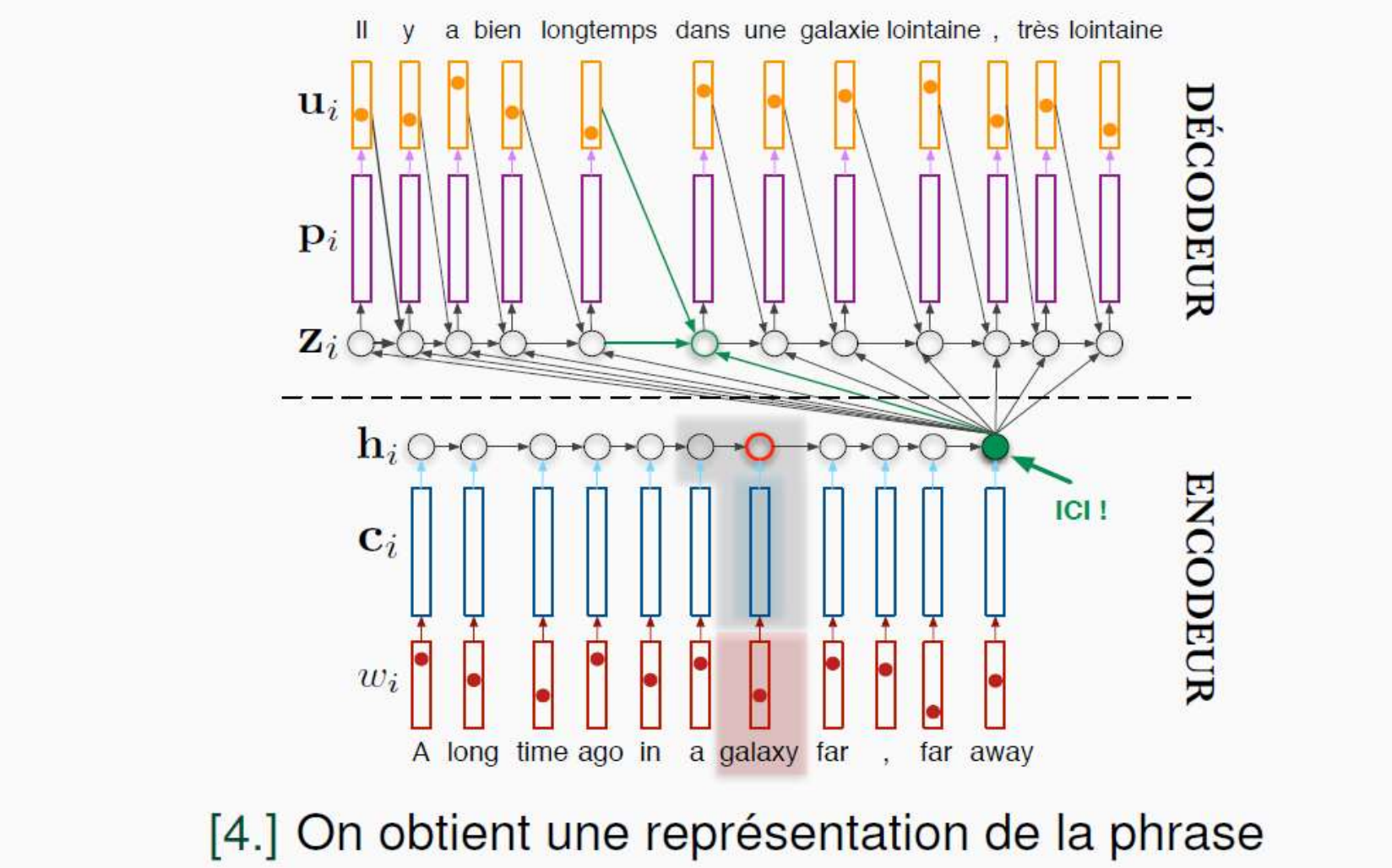
[2.] Projection du vecteur 1hot dans l'espace continu : embedding

Architecture des modèles NMT: Encodeur- Décodeur

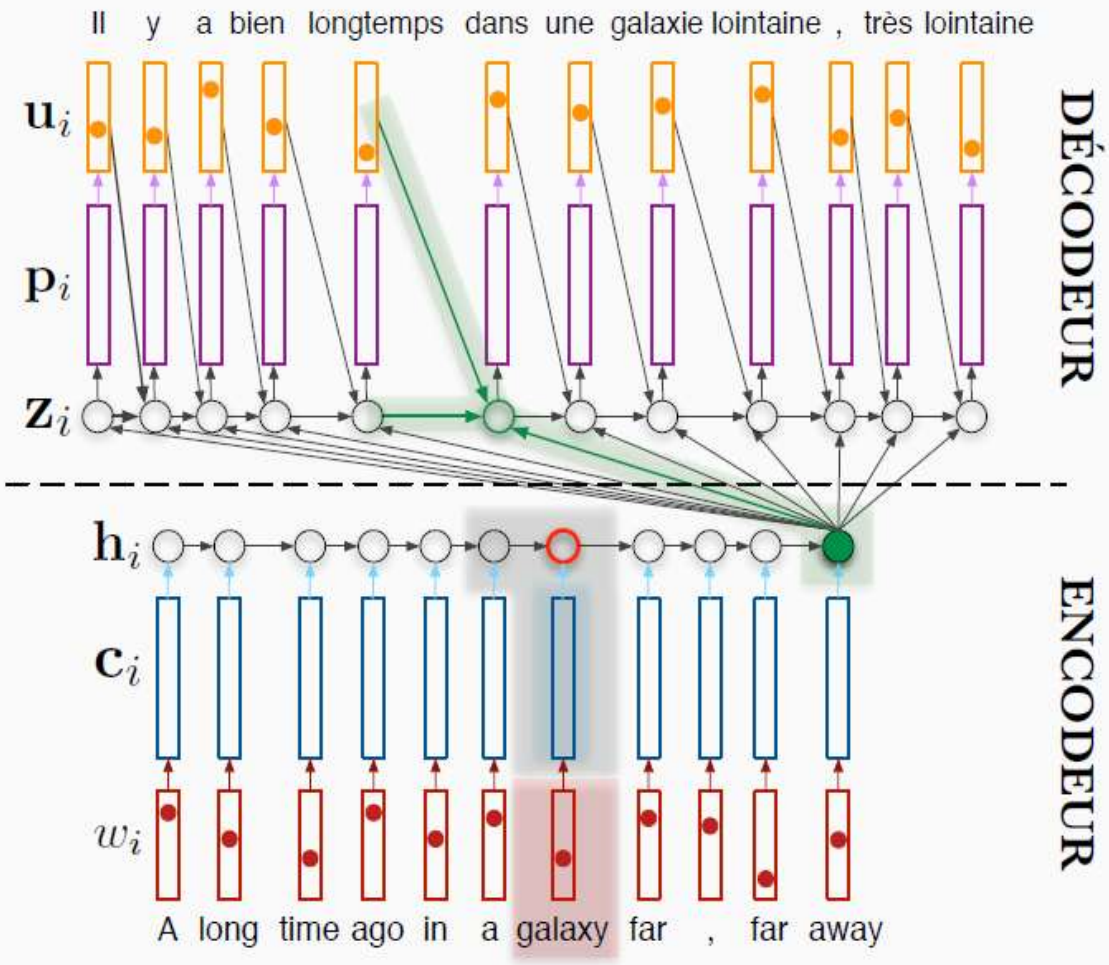


[3.] Mise-à-jour incrémentale de l'état caché de l'unité récurrente de *l'encodeur*

Architecture des modèles NMT: Encodeur- Décodeur

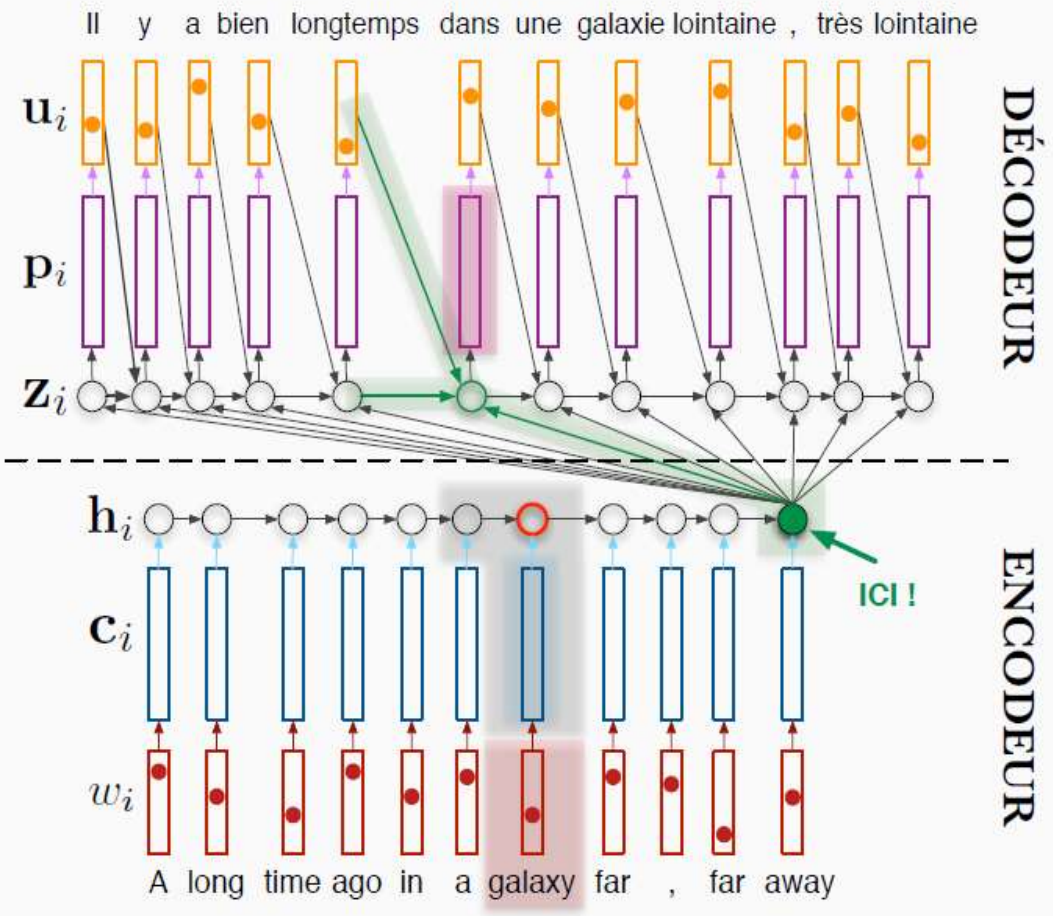


Architecture des modèles NMT: Encodeur- Décodeur



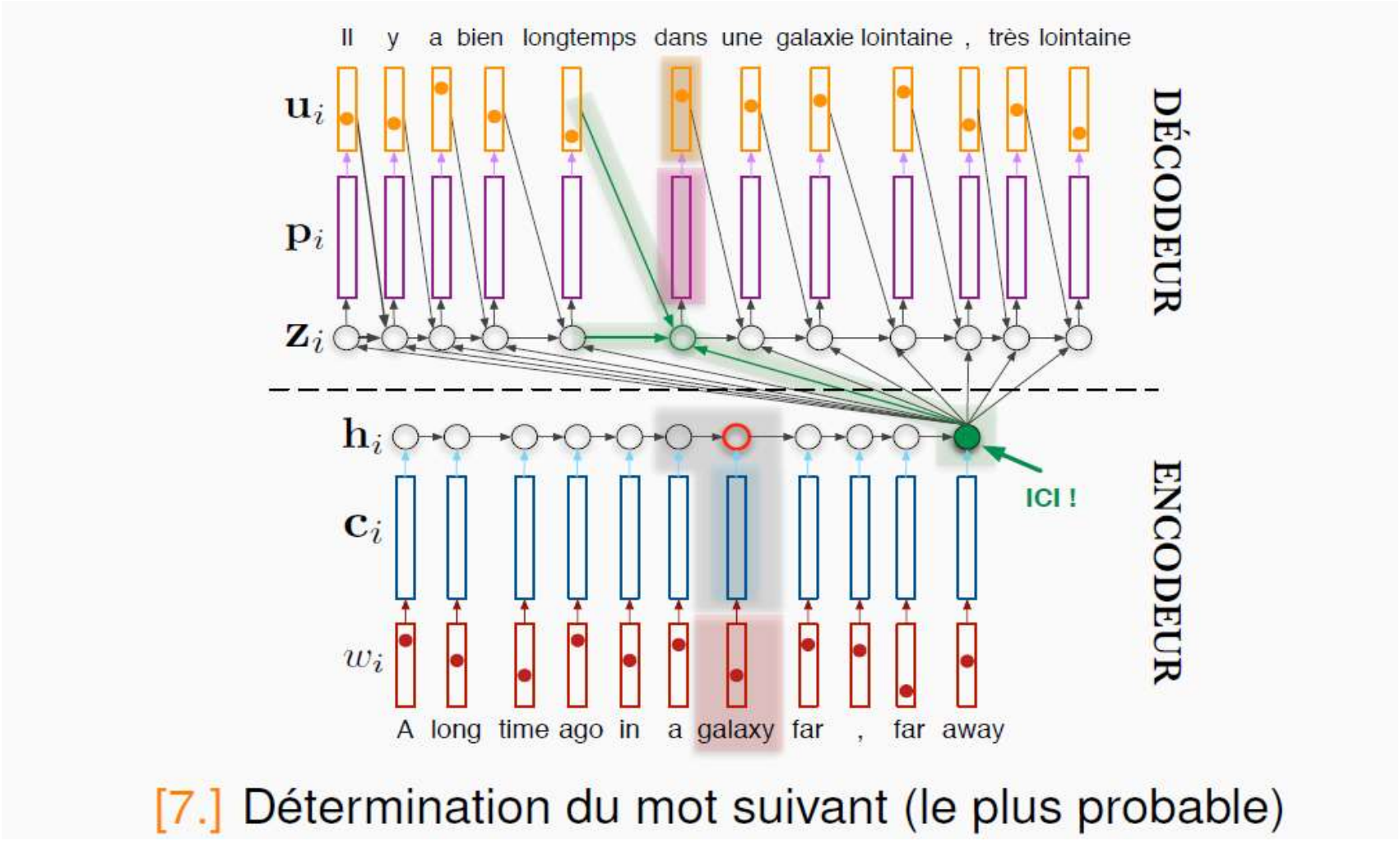
[5.] Mise-à-jour incrémentale de l'état caché de l'unité récurrente du *décodeur*

Architecture des modèles NMT: Encodeur- Décodeur



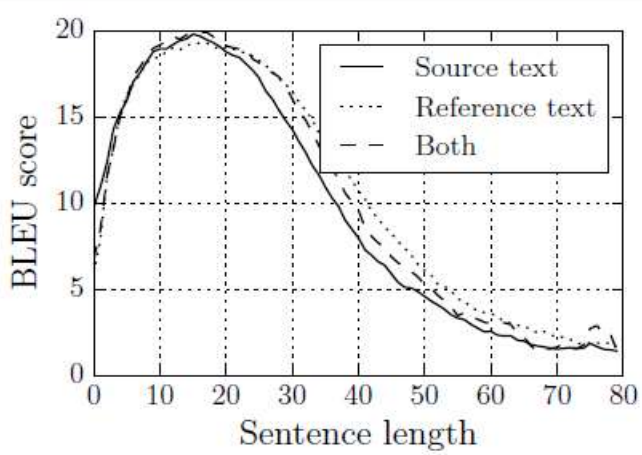
[6.] Calcul de la distribution de probabilité pour tous les mots suivants

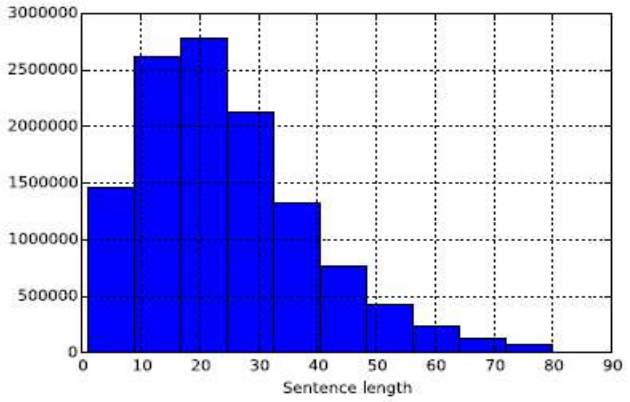
Architecture des modèles NMT: Encodeur- Décodeur



[7.] Détermination du mot suivant (le plus probable)

Modèles NMT: Résultats

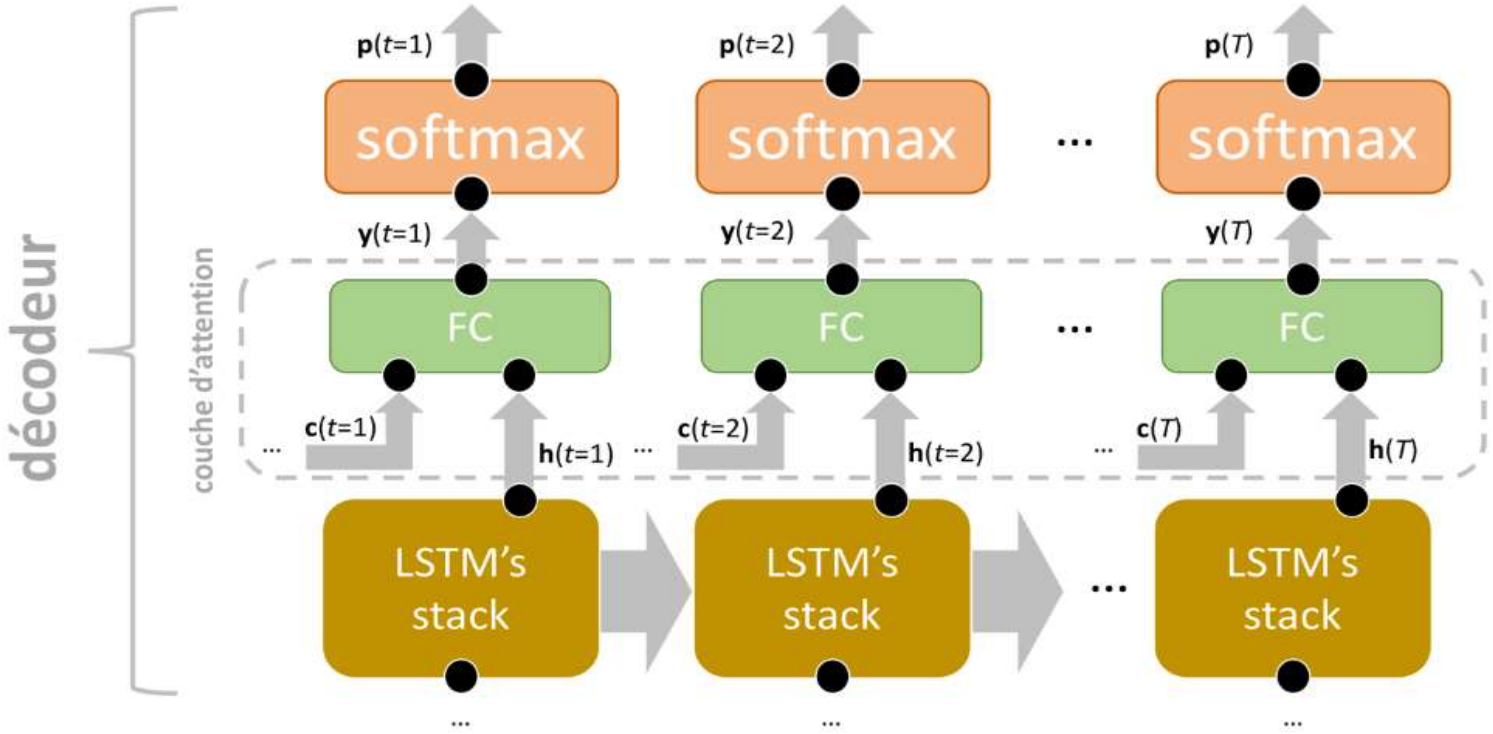




→ Le score de traduction décroit avec la taille de la phrase !

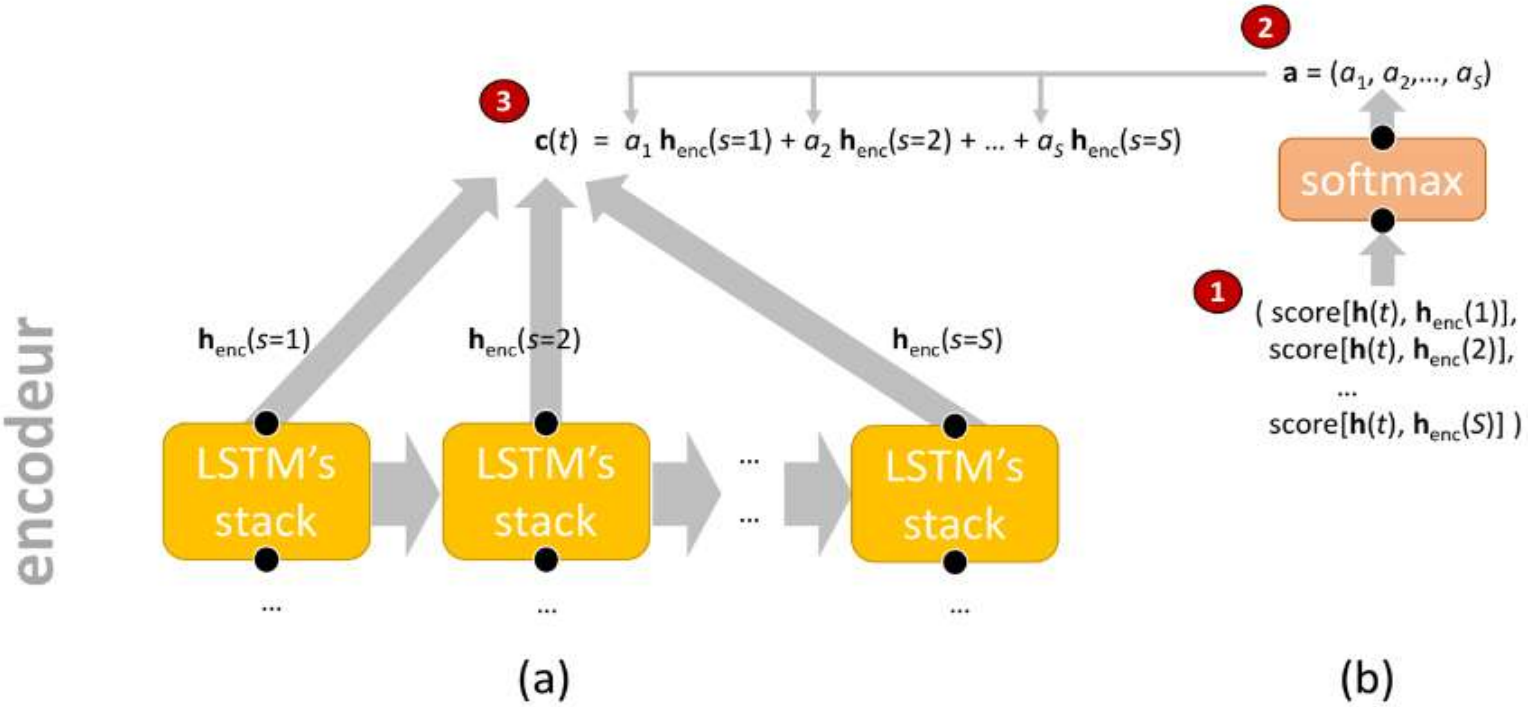
- Comment expliquer cela ?
  1. Pas assez de données dans le corpus d'entraînement ?
  2. Difficile de générer une phrase longue cohérente ?
  3. Vecteur de taille fixe insuffisant pour encoder une longue phrase ?

Architecture des modèles NMT: Encodeur-Décodeur avec des LSTMs et mécanisme d'attention



Une couche d'attention intercalée entre la dernière couche de LSTM et les softmax permet de tenir compte à la fois du contexte h(t) local au décodeur et d'un contexte c(t) calculé à partir de la phrase source

Architecture des modèles NMT: Encodeur-Décodeur avec des LSTMs et mécanisme d'attention



Le vecteur de contexte c(t) utilisé est une somme pondérée des S vecteurs de contexte côté encodeur, il est calculé en 3 étapes: (1) on calcule des scores de proximité entre le contexte h(t) coté décodeur et les S contextes $h_{enc}(s)$ associés à chaque mot côté encodeur, (2) on transforme ces scores en probabilités a_s avec un softmax et (3) on utilise ces probabilités a_s pour calculer c(t) comme une somme pondérée des S contextes $h_{enc}(s)$

Modèles NMT: Bilan

- Utilisation d'un décodeur à faisceau (*beam search*)

→ Exploration d'un espace de recherche plus important

- Utilisation de GPUs

→ Calcul parallèle inhérent à la structure de ces processeurs

→ Multiplication de matrices

- Modèles entraînés de manière incrémentale

→ raffinement (*fine tuning*)

→ adaptation (en utilisant des données du domaine)

Modèles NMT: Bilan

Séquence vers séquence

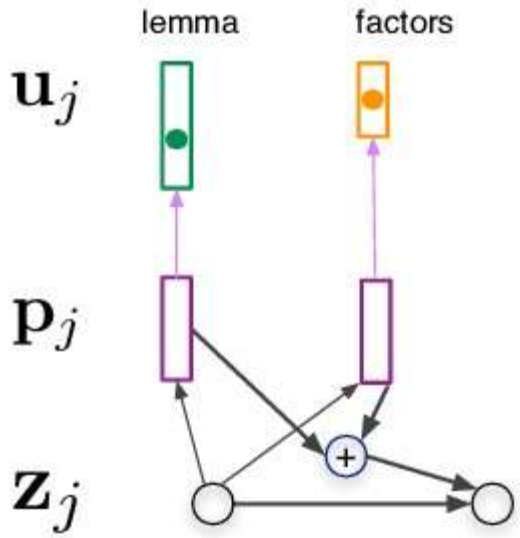
- Utilisation de réseaux de neurones récurrents (bidirectionnels)
- Utilisation d'unité à portes (LSTM ou GRU)

$\rightarrow$ Apprendre à mémoriser/oublier

$\Rightarrow$ Approche encodeur/décodeur

- Encodeur
  - Compresser l'entrée dans un vecteur de taille fixe
  - Avec attention : sorte de vecteur *moyen* obtenu par pondération des annotations
- Décodeur
  - Générer une séquence à partir de la représentation compressée

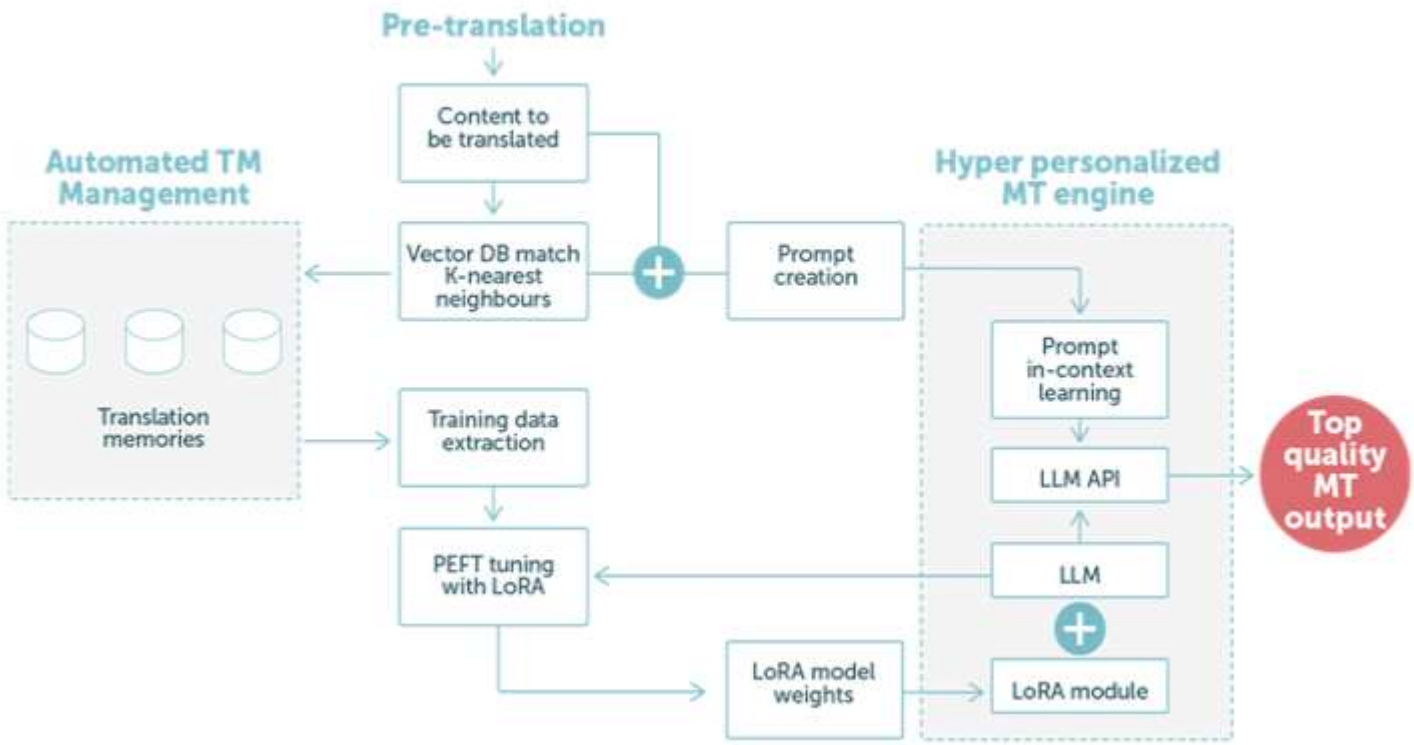
Les variantes des systèmes NMT: La traduction neuronale factorisée



Deux sorties :

- **Le Lemme**
- **La concaténation des différents facteurs considérés (nombre, genre, etc,)**

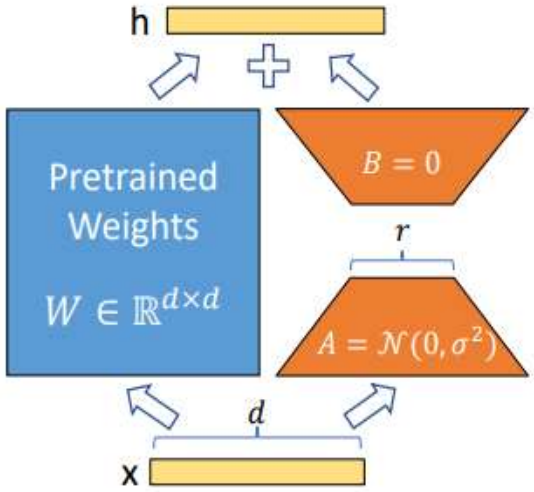
NMT utilisant un grand modèle de langue (LLM)



Combinaison de personnalisations LoRA et d'invites d'apprentissage contextuelles optimisées

- Lorsque l'invite spécialement conçue est traitée par le LLM, les poids personnalisés du module LoRA contribuent à un résultat de traduction automatique de qualité supérieure.
- Une fois cette étape terminée, le résultat passe automatiquement à l'étape suivante du processus: post-édition avec un expert humain pour une qualité finale maximale.

Fine-tuning un LLM avec LoRA



LoRA (Low-Rank Adaptation of LLMs): L'entrainement est réalisé uniquement sur A et B

Model & Method	# Trainable Parameters	E2E NLG Challenge BLEU	NIST	MET	ROUGE-L	CIDEr
GPT-2 M (FT)*	354.92M	68.2	8.62	46.2	71.0	2.47
GPT-2 M (Adapter^{L})*	0.37M	66.3	8.41	45.0	69.8	2.40
GPT-2 M (Adapter^{L})*	11.09M	68.9	8.71	46.1	71.3	2.47
GPT-2 M (Adapter^{H})	11.09M	$67.3_{\pm.6}$	$8.50_{\pm.07}$	$46.0_{\pm.2}$	$70.7_{\pm.2}$	$2.44_{\pm.01}$
GPT-2 M (FT^{Top2})*	25.19M	68.1	8.59	46.0	70.8	2.41
GPT-2 M (PreLayer)*	0.35M	69.7	8.81	46.1	71.4	2.49
GPT-2 M (LoRA)	0.35M	$\mathbf{70.4}_{\pm.1}$	$\mathbf{8.85}_{\pm.02}$	$\mathbf{46.8}_{\pm.2}$	$\mathbf{71.8}_{\pm.1}$	$\mathbf{2.53}_{\pm.02}$
GPT-2 L (FT)*	774.03M	68.5	8.78	46.0	69.9	2.45
GPT-2 L (Adapter^{L})	0.88M	$69.1_{\pm.1}$	$8.68_{\pm.03}$	$46.3_{\pm.0}$	$71.4_{\pm.2}$	$\mathbf{2.49}_{\pm.0}$
GPT-2 L (Adapter^{L})	23.00M	$68.9_{\pm.3}$	$8.70_{\pm.04}$	$46.1_{\pm.1}$	$71.3_{\pm.2}$	$2.45_{\pm.02}$
GPT-2 L (PreLayer)*	0.77M	70.3	8.85	46.2	71.7	2.47
GPT-2 L (LoRA)	0.77M	$\mathbf{70.4}_{\pm.1}$	$\mathbf{8.89}_{\pm.02}$	$\mathbf{46.8}_{\pm.2}$	$\mathbf{72.0}_{\pm.2}$	$2.47_{\pm.02}$

Quelques liens pour des outils de TA

- **Traduction à base de règles - Apertium**

https://github.com/apertium

- **Traduction statistique - Moses**

https://github.com/tfeng/moses

- **Traduction neuronale - OpenNMT**

https://opennmt.net/

- **Traduction neuronale - Fairseq**

https://github.com/facebookresearch/fairseq

- **Traduction neuronale - Transformers**

https://github.com/jeffprosise/Applied-Machine-Learning/blob/main/Chapter%2013/Neural%20Machine%20Translation%20(Transformer).ipynb

- **Traduction neuronale avec des LLMs**

https://github.com/ymoslem/Adaptive-MT-LLM