

# Correction automatique des productions écrites d'apprenants de FLE: introduction et applications

Xiaou Wang, UCLouvain  
[xiaoou.wang@uclouvain.be](mailto:xiaoou.wang@uclouvain.be)

- Correction automatique des textes
- Intuitions
- Traduction automatique neuronale (TAN)
- RNN, attention, Transformer
- structures de données et algorithme
- programmation dynamique
- Annotations morphosyntaxiques
- Erreurs des apprenants de langue étrangère
- Evaluation automatique de productions écrites (Automated Essay Scoring, AES)
- etc.....

# Correction automatique

“Les erreurs n'ont pas toujours la forme d'erreurs :  
un énoncé correct peut cacher une erreur.”

(Martine Marquilló Larruy, (2003) *L'interprétation de l'erreur*)

- erreurs orthographiques (La tlévision que j'ai achetée)
- erreurs grammaticales (La télévision que j'ai acheté)

# Correction automatique

"Les erreurs n'ont pas toujours la forme d'erreurs :  
un énoncé correct peut cacher une erreur."

(Martine Marquilló Larruy, (2003) *L'interprétation de l'erreur*)

- erreurs d'usage

- (1) le lion pensif
- (2) le fromage pensif (pas d'erreur grammaticale mais sémantique)
- (3) Non naturel ? (Film theater, collocation, ni erreur grammaticale si sémantique mais non conforme à l'usage)

38% and 46% of all verb-object constructions in English journalistic prose (e.g., hold an election [organiser une élection], throw a party [faire une fête]) are collocations. (Cowie, 1991)

- erreurs stylistiques (mot familier dans un texte au registre formel)

# Erreur orthographique

- La correction orthographique (aussi appelée correction lexicale parfois, non-word error) vise principalement à corriger des formes qui ne respectent pas la règle d'écriture d'un mot (oisea pour oiseau)
- Premiers travaux dès les années 1960 selon Kukich (1992)

# Correction orthographique

- Premier article (Blair, 1960)

An  $r$ -letter abbreviation of an  $n$ -letter word can be produced by deleting those  $n - r$  letters which are least important in the identification of the word. The problem of producing an adequate abbreviation is, in application, that of deciding which letters in a word are the least important in determining its meaning. Information theorists assume that the information conveyed by a “message” is inversely proportional to its *a priori* probability of occurrence. One can apply this idea by eliminating the  $n - r$  letters in the order of their expected frequency; we tried this but found that even better results can be obtained by using the “frequency” of their occurrence as errors. An empirically constructed ap-

- Blair, C. R. (1960). A program for correcting spelling errors. *Information and Control*, 3(1), 60–67.

# Correction orthographique

- Premier article (Blair, 1960)

as “xpnn” for exponent. Clearly weight must also be given to the position of the letter in the word. The first letter is of greatest importance, and, all other things being equal, the last letter is second in importance, followed by the second letter, the next to last letter, etc. That is, if we reorder the letters in this fashion, the desirability of rejecting a letter in a given position is an increasing, monotonic function of the new posi-

- Blair, C. R. (1960). A program for correcting spelling errors. *Information and Control*, 3(1), 60–67.

# Correction orthographique

- Premier article (Blair, 1960), desirability of deletion

Letter	Score	Letter	Score	Position	Score	Position	Score
A	5	N	3				
B	1	O	4	1	0	9	5
C	5	P	3	2	1	10	5
D	0	Q	0	3	2	11	6
E	7	R	4	4	3	12	6
F	1	S	5	5	4	13	6
G	2	T	3	6	4	14	6
H	5	U	4	7	5	15	6
I	6	V	1				
J	0	W	1				
K	1	X	0				
L	5	Y	2				
M	1	Z	1	8	5	16 up	7

# Correction orthographique

- Premier article (Blair, 1960)
- By assuming that the name and position of a letter independently determine the desirability of rejecting it, one can form an r-letter abbreviation by **deleting** the n - r letters which have the largest product
- Before it is asked to correct misspelled words, the machine must compute and store a short (we need 4 letters) abbreviation of each correctly spelled word in the vocabulary.

## EXAMPLE

A	B	S	O	R	B	E	N	T
5	1	5	4	4	1	7	3	3.
0	2	4	5	5	5	4	3	1
5	3	9	9	9	6	11	6	4
*	*	*	*	*	*	*	*	
	A	B	B	B	T			

Letter score  
Position score  
Sum of scores  
Delete  
Abbreviation

A	B	S	O	R	B	A	N	T
5	1	5	4	4	1	5	3	3
0	2	4	5	5	5	4	3	1
5	3	9	9	9	6	9	6	4
*	*	*	*	*	*	*	*	*
	A	B	B	B	T			

- Blair, C. R. (1960). A program for correcting spelling errors. Information and Control, 3(1), 60–67.

# Correction orthographique

- Premier article (Blair, 1960)

By assuming that the name and position of a letter independently determine the desirability of rejecting it, one can form an r-letter abbreviation by deleting the n-r letters which have the largest product<sup>2</sup>

Before it is asked to correct misspelled words, the machine must compute and store a short (we used 4 letters) abbreviation of each correctly spelled word in the vocabulary.

## EXAMPLE

A	B	S	O	R	B	E	N	T
5	1	5	4	4	1	7	3	3.
0	2	4	5	5	5	4	3	1
5	3	9	9	9	6	11	6	4
*	*	*	*	*	*	*	*	
	A	B	B	B	T			

Letter score	5	1	5	4	4	1	5	3	3
Position score	0	2	4	5	5	5	4	3	1
Sum of scores	5	3	9	9	9	6	9	6	4
Delete	*	*	*	*	*	*	*	*	*
Abbreviation	A	B	B	B	T				

If multiple words, compares longer abbreviations of this input word with longer abbreviations of the vocabulary entries it matched until a unique one has been selected.

# Correction orthographique

The association of common misspellings with their correctly spelled equivalents is illustrated in Table III. The program correctly identified 89 of the 117 misspelled words (3 required longer abbreviations) while incorrectly identifying ~~comt~~.

Un système à base de règles typique avec

1, ressources (correct spelling)  
2, règles

Correct spelling	Abbreviations		Incorrect spelling
ABSORBENT	ABBT	=	ABSORBANT
ABSORPTION	ABON	=	ABSORBTION
ACCOMMODATE	AMDT	=	ACCOMODATE
ACQUIESCE	ACQC	=	AQUIESE
ANALYZE	ANYZ	=	ANALIZE
ANTARCTIC	ANTC	=	ANTARTIC
ASININE	ASNN	=	ASSININE
ASSISTANCE	ASTN	=	ASSISTENCE
AUXILIARY	AUXY	=	AUXILLARY
BANANA	BANA	=	BANANNA
BANKRUPTCY	BAKY	=	BANKRUPCY
BRETHREN	BRTN	=	BRETERHEN
BRITAIN	BRTN	=	BRITIAN
BUOYANCY	BUYY	=	BOUYANCY
CATEGORY	CATY	=	CATAGOREY
CHAUFFEUR	CFFR	=	CHAUFFUER
CHIMNEYS	CMYS	=	CHIMNIES
COLISEUM	COUM	=	COLOSIMUM
COLOSSAL	COAL	=	COLLOSOAL
COMMITMENT	COMT	=	COMMITTMENT

# Correction orthographique

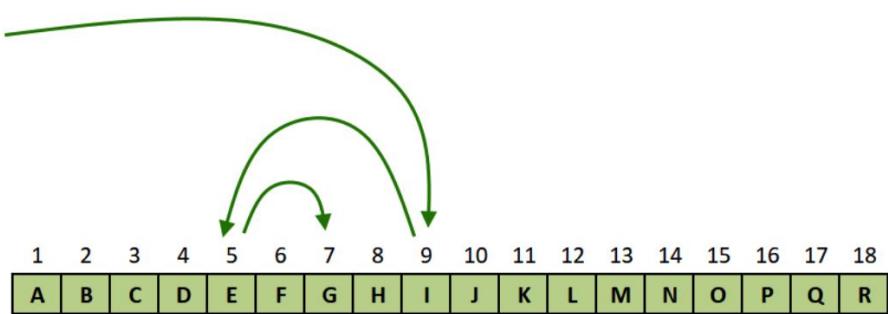
Pourquoi ne pas vérifier directement si une forme existe dans un vocabulaire  $\{v\}$

1. Stockage (ok)
2. Performance (?)
3. Nouveaux mots (partiellement résolu, ajout de liste de mots personnalisée)
4. Correction ?

# Correction orthographique

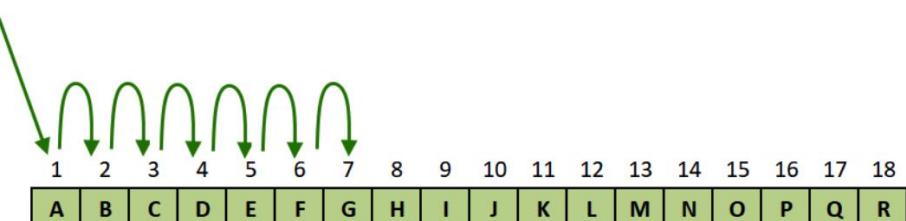
Pourquoi ne pas vérifier directement si une forme existe dans un vocabulaire  $\{v\}$

## 1. Performance (structure de données et algorithme)



Binary Search - Find 'G' in sorted list A-R

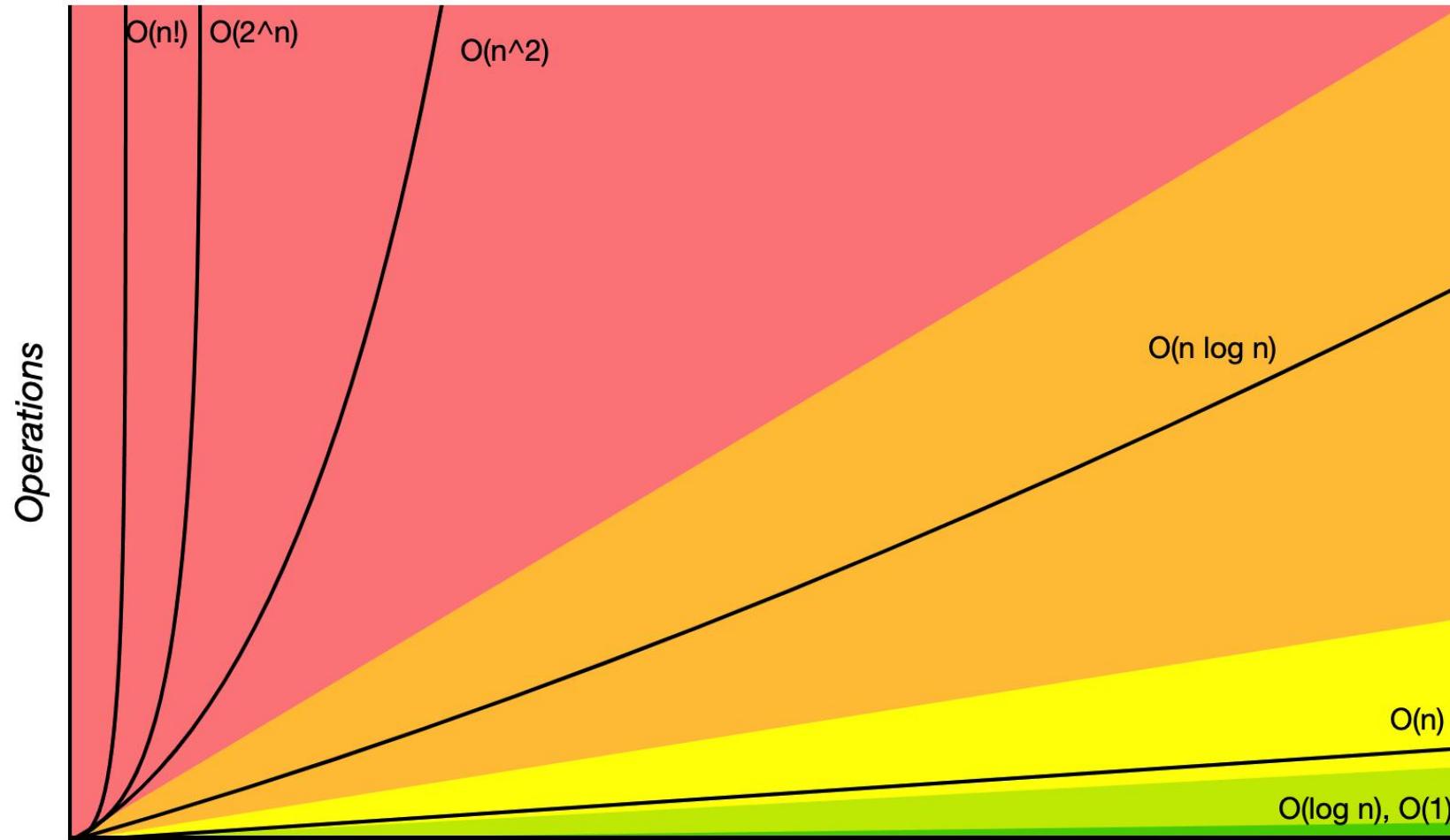
best case:  $O(1)$   
average case:  $O(N)$  pour la recherche linéaire,  $O(\log_2 N)$  dans le cas de la recherche binaire  
worst case :  $O(N)$  pour la recherche linéaire,  $O(\log_2 N)$  pour la recherche binaire



Linear Search - Find 'G' in sorted list A-R

# Big-O Complexity Chart

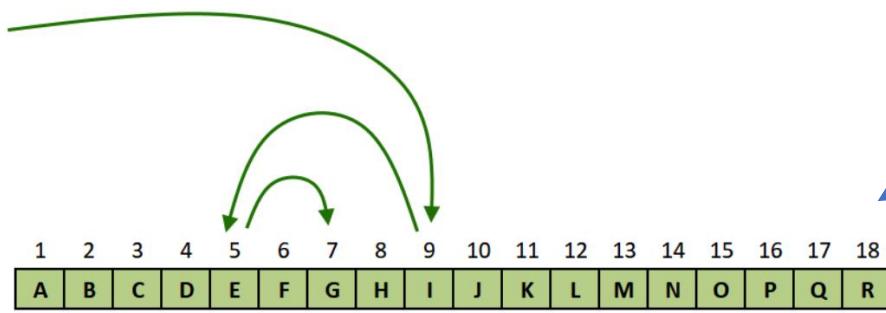
Horrible   Bad   Fair   Good   Excellent



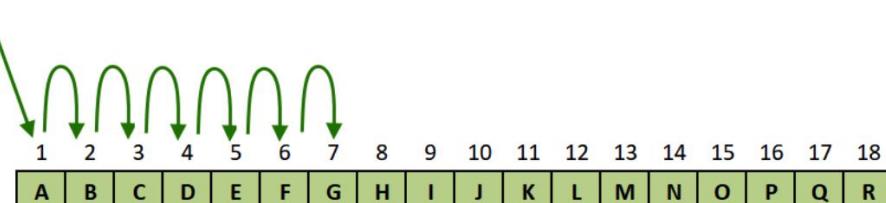
# Correction orthographique

Pourquoi ne pas vérifier directement si une forme existe dans un vocabulaire  $\{v\}$

1. Performance (structure de données et algorithme)      triche ? cette liste est déjà triée !



Binary Search - Find 'G' in sorted list A-R



Linear Search - Find 'G' in sorted list A-R

algorithme de tri (le tri ne se fait qu'une fois)

Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	
Quicksort	$\Omega(n \ log(n))$	$\Theta(n \ log(n))$	$\Theta(n^2)$	$O(\log(n))$
Mergesort	$\Omega(n \ log(n))$	$\Theta(n \ log(n))$	$\Theta(n \ log(n))$	$O(n)$
Timsort	$\Omega(n)$	$\Theta(n \ log(n))$	$\Theta(n \ log(n))$	$O(n)$
Heapsort	$\Omega(n \ log(n))$	$\Theta(n \ log(n))$	$\Theta(n \ log(n))$	$O(1)$
Bubble Sort	$\Omega(n)$	$\Theta(n^2)$	$\Theta(n^2)$	$O(1)$
Insertion Sort	$\Omega(n)$	$\Theta(n^2)$	$\Theta(n^2)$	$O(1)$
Selection Sort	$\Omega(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	$O(1)$
Tree Sort	$\Omega(n \ log(n))$	$\Theta(n \ log(n))$	$\Theta(n \ log(n))$	$O(n)$
Shell Sort	$\Omega(n \ log(n))$	$\Theta(n(\log(n))^2)$	$\Theta(n(\log(n))^2)$	$O(1)$
Bucket Sort	$\Omega(n+k)$	$\Theta(n+k)$	$\Theta(n^2)$	$O(n)$
Radix Sort	$\Omega(nk)$	$\Theta(nk)$	$\Theta(nk)$	$O(n+k)$
Counting Sort	$\Omega(n+k)$	$\Theta(n+k)$	$\Theta(n+k)$	$O(k)$
Cubesort	$\Omega(n)$	$\Theta(n \ log(n))$	$\Theta(n \ log(n))$	$O(n)$

# Correction orthographique

Pourquoi ne pas vérifier directement si une forme existe dans un vocabulaire  $\{v\}$

1. Performance (structure de données et algorithme)

```
lst = [i for i in range(1,10000000)]  
  
def linear_search(arr,x):  
    for i in range(len(arr)):  
        if arr[i] == x:  
            return i  
    return -1
```

```
def binary_search(arr, low, high, x):  
    if high >= low:  
        mid = (high + low) // 2  
        if arr[mid] == x:  
            return mid  
        elif arr[mid] > x:  
            return binary_search(arr, low, mid - 1, x)  
        else:  
            return binary_search(arr, mid + 1, high, x)  
    else:  
        return -1
```

# Correction orthographique

Pourquoi ne pas vérifier directement si une forme existe dans un vocabulaire  $\{v\}$

1. Performance (structure de données et algorithme)

```
%%timeit -n 1000
binary_search(lst, 0, len(lst)-1, 500)
```

✓ 0.7s

8 µs ± 642 ns per loop (mean ± std. dev. of 7 runs, 1000 loops each)

```
%%timeit -n 1000
linear_search(lst, 500)
```

✓ 0.3s

26.4 µs ± 1.95 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)

# Correction orthographique

Pourquoi ne pas vérifier directement si une forme existe dans un vocabulaire  $\{v\}$

- Pas de correction, mais utile quand même (real-time detection, user interaction)

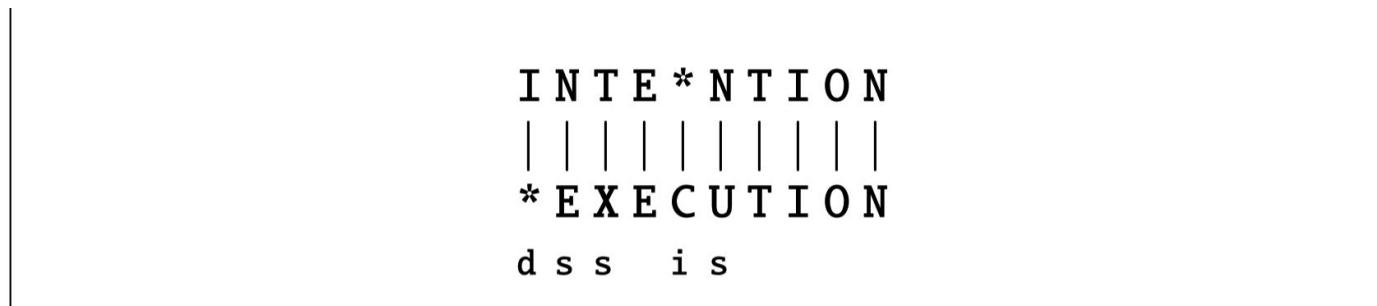
# Correction orthographique

Pourquoi ne pas vérifier directement si une forme existe dans un vocabulaire  $\{v\}$

## 4. Correction ?

Distance de Levenshtein (Levenshtein, 1966)

Définition de la distance (passer d'un mot à l'autre avec trois types d'opérations)



**Figure 2.14** Representing the minimum edit distance between two strings as an **alignment**. The final row gives the operation list for converting the top string into the bottom string: **d** for deletion, **s** for substitution, **i** for insertion.

# Correction orthographique

Pourquoi ne pas vérifier directement si une forme existe dans un vocabulaire  $\{v\}$

4. Correction ?

Distance de Levenshtein (Levenshtein, 1966)

Définition de la distance (passer d'un mot à l'autre avec trois types d'opérations)

Trouver la distance minimum, Algorithme (Wagner and Fischer 1974)

$$D[i, j] = \min \begin{cases} D[i - 1, j] + \text{del-cost}(\text{source}[i]) \\ D[i, j - 1] + \text{ins-cost}(\text{target}[j]) \\ D[i - 1, j - 1] + \text{sub-cost}(\text{source}[i], \text{target}[j]) \end{cases}$$

delete, insertion, insertion

# Correction orthographique

Pourquoi ne pas vérifier directement si une forme existe dans un vocabulaire  $\{v\}$

$$D[i, j] = \min \begin{cases} D[i - 1, j] + \text{del-cost}(\text{source}[i]) \\ D[i, j - 1] + \text{ins-cost}(\text{target}[j]) \\ D[i - 1, j - 1] + \text{sub-cost}(\text{source}[i], \text{target}[j]) \end{cases}$$

$\text{del} = 1$   
 $\text{ins} = 1$   
 $\text{sub} = 2$  (0 if  
the same  
letter)

Src \ Tar	#	e	x	e	c	u	t	i	o	n
#	0	1	2	3	4	5	6	7	8	9
i	1	2	3	4	5	6	7	6	7	8
n	2	3	4	5	6	7	8	7	8	7
t	3	4	5	6	7	8	7	8	9	8
e	4	3	4	5	6	7	8	9	10	9
n	5	4	5	6	7	8	9	10	11	10
t	6	5	6	7	8	9	8	9	10	11
i	7	6	7	8	9	10	9	8	9	10
o	8	7	8	9	10	11	10	9	8	9
n	9	8	9	10	11	12	11	10	9	8

programmation  
dynamique

# Correction orthographique

Pourquoi ne pas vérifier directement si une forme existe dans un vocabulaire {v}

$$D[i, j] = \min \begin{cases} D[i - 1, j] + \text{del-cost}(\text{source}[i]) & \text{del} = 1 \\ D[i, j - 1] + \text{ins-cost}(\text{target}[j]) & \text{ins} = 1 \\ D[i - 1, j - 1] + \text{sub-cost}(\text{source}[i], \text{target}[j]) & \text{sub} = 2 \end{cases}$$

Exemple concret :

```
cities["Tuscon"] cities["Pittsburg"] cities["Cincinnati"]
cities["Albequerque"]
```

```
cities = {"Pittsburgh": "Pennsylvania",
          "Tucson": "Arizona",
          "Cincinnati": "Ohio",
          "Albuquerque": "New Mexico",
          "Culpeper": "Virginia",
          "Asheville": "North Carolina",
          "Worcester": "Massachusetts",
          "Manhattan": "New York",
          "Phoenix": "Arizona",
          "Niagara Falls": "New York"}
```

# Correction orthographique

Pourquoi ne pas vérifier directement si une forme existe dans un vocabulaire {v}

$$D[i, j] = \min \begin{cases} D[i - 1, j] + \text{del-cost}(\text{source}[i]) \\ D[i, j - 1] + \text{ins-cost}(\text{target}[j]) \\ D[i - 1, j - 1] + \text{sub-cost}(\text{source}[i], \text{target}[j]) \end{cases}$$

```
def LD(s, t):  
    if s == "":  
        return len(t)  
    if t == "":  
        return len(s)  
    if s[-1] == t[-1]:  
        cost = 0  
    else:  
        cost = 1  
    res = min([LD(s[:-1], t) + 1,  
              LD(s, t[:-1]) + 1,  
              LD(s[:-1], t[:-1]) + cost])  
  
    return res
```

Src\Tar	#	e	x	e	c	u	t	i	o	n
#	0	1	2	3	4	5	6	7	8	9
i	1	2	3	4	5	6	7	6	7	8
n	2	3	4	5	6	7	8	7	8	7
t	3	4	5	6	7	8	7	8	9	8
e	4	3	4	5	6	7	8	9	10	9
n	5	4	5	6	7	8	9	10	11	10
t	6	5	6	7	8	9	8	9	10	11
i	7	6	7	8	9	10	9	8	9	10
o	8	7	8	9	10	11	10	9	8	9
n	9	8	9	10	11	12	11	10	9	8

# Correction orthographique

Pourquoi ne pas vérifier directement si une forme existe dans un vocabulaire {v}

$$D[i, j] = \min \begin{cases} D[i - 1, j] + \text{del-cost}(\text{source}[i]) \\ D[i, j - 1] + \text{ins-cost}(\text{target}[j]) \\ D[i - 1, j - 1] + \text{sub-cost}(\text{source}[i], \text{target}[j]) \end{cases}$$

del = 1  
ins = 1  
sub = 2

```
cities = {"Pittsburgh": "Pennsylvania",
          "Tucson": "Arizona",
          "Cincinnati": "Ohio",
          "Albuquerque": "New Mexico",
          "Culpeper": "Virginia",
          "Asheville": "North Carolina",
          "Worcester": "Massachusetts",
          "Manhattan": "New York",
          "Phoenix": "Arizona",
          "Niagara Falls": "New York"}
```

```
def LD(s, t):
    if s == "":
        return len(t)
    if t == "":
        return len(s)
    if s[-1] == t[-1]:
        cost = 0
    else:
        cost = 2
    res = min([LD(s[:-1], t)+1,
              LD(s, t[:-1])+1,
              LD(s[:-1], t[:-1]) + cost])
    return res
```

```
cities = {"Pittsburgh": "Pennsylvania",
          "Tucson": "Arizona",
          "Cincinnati": "Ohio",
          "Albuquerque": "New Mexico",
          "Culpeper": "Virginia",
          "Asheville": "North Carolina",
          "Worcester": "Massachusetts",
          "Manhattan": "New York",
          "Phoenix": "Arizona",
          "Niagara Falls": "New York"}
for city in cities:
    print(LD("Tuscon", city), end=",")
14,2,12,15,12,13,13,13,9,17,
```

# Correction orthographique

Pourquoi ne pas vérifier directement si une forme existe dans un vocabulaire {v}

$$D[i, j] = \min \begin{cases} D[i - 1, j] + \text{del-cost}(\text{source}[i]) & \text{del} = 1 \\ D[i, j - 1] + \text{ins-cost}(\text{target}[j]) & \text{ins} = 1 \\ D[i - 1, j - 1] + \text{sub-cost}(\text{source}[i], \text{target}[j]) & \text{sub} = 2 \end{cases}$$

Modélisation des erreurs à base de la distance d'édition avec l'intégration de plus en plus de paramètres (proximités phonétiques (Veronis, 1988), proximité des touches sur le clavier (Sagot & Boullier, 2008)

oiso /wazo/ -> oiseau /wazo/  
déseaulé /dezole/ -> désolé /dezole/

s est plus facile à confondre avec la touche |d| que la touche |i|...

# Correction orthographique et grammaticale

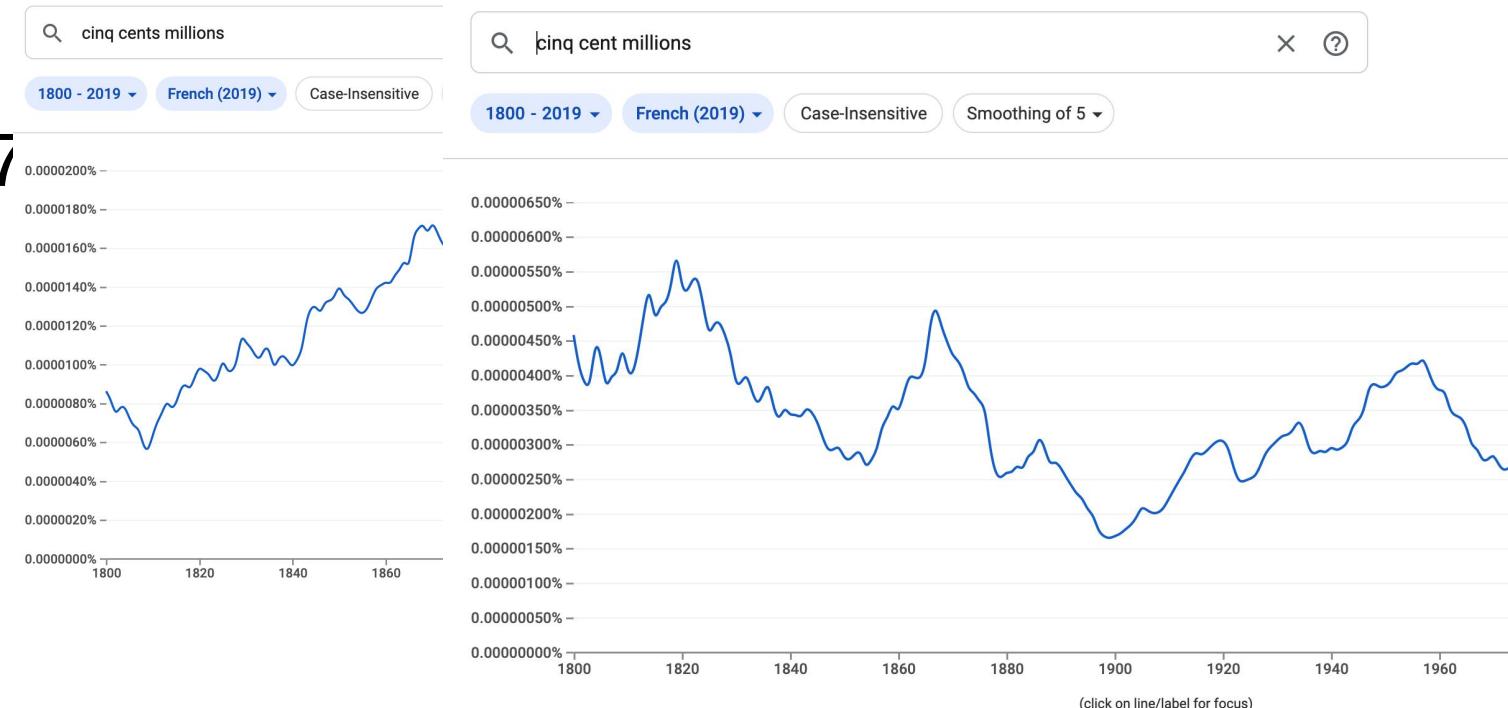
- Et le contexte ? -> a peace of cake (real-word errors)
- création d'un confusion set avec la distance de levenshtein et d'autres critères (homophone etc...) (Golding, 1996; Golding & Roth, 1996)
- piece -> peace (homophone), among -> between (grammaire), form -> from (distance de levenshtein)
- {sang, cent, cents, cen}
- Comment déterminer quel est le bon mot ? Apprentissage statistique (utilisation de corpus)

# Correction orthographique et grammaticale

- Comment déterminer quel est le bon mot ? Apprentissage statistique (utilisation de corpus) -> Exemple

- n-gramme (Carlson 2007)

- cinq cent millions
- cinq cents cinquante



# Correction orthographique et grammaticale

- La prise en compte du contexte  $\approx$  erreurs grammaticales

beaucoup d'autres approches...

approches par règles : if then

# Correction orthographique et grammaticale

- La prise en compte du contexte  $\approx$  erreurs grammaticales

approches par règles : if then

The product is not only cheap but also very good!  
I always wanted this feature badly!



```
def get_sentiment(text):  
    if 'good' in text:  
        return 1  
    elif 'bad' in text:  
        return -1  
    return 0
```



```
def get_sentiment(text):  
    if 'good' in text:  
        return 1  
    elif 'bad' in text:  
        return -1  
    if 'not' in text or "n't" in text:  
        sentiment *= -1  
    return 0
```

it's good

it's not good

# Correction orthographique et grammaticale

- La prise en compte du contexte  $\approx$  erreurs grammaticales

approches par règles : if then

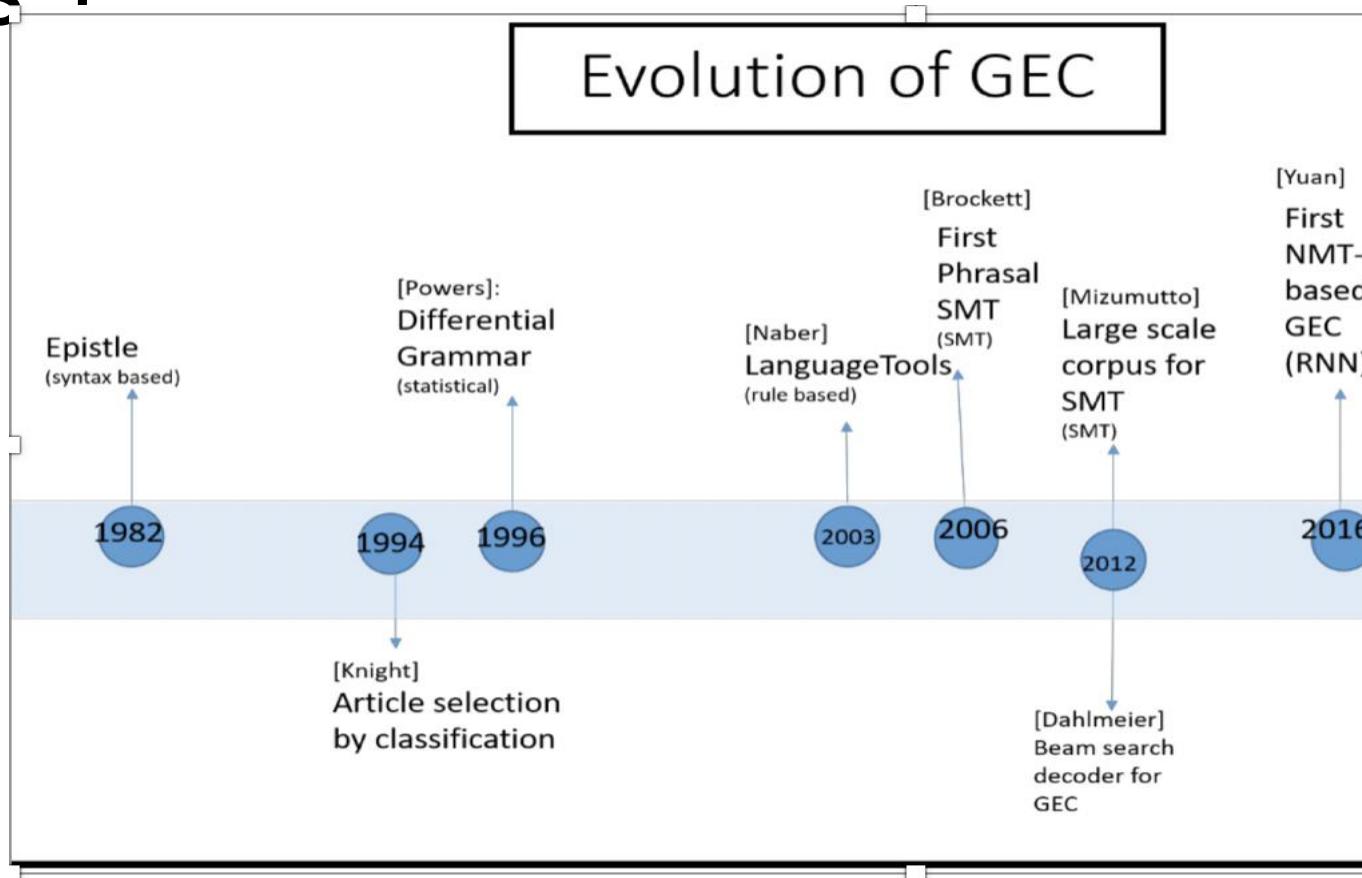
The product is not only cheap but also very good!  
I always wanted this feature badly!

Création automatique des règles (Mangu et al. 1997)

Classification automatique des erreurs à partir des corpus annotés  
(erreurs limitées car nécessitant des corpus annotés en erreurs,  
Rozovskaya 2014)

# Correction automatique et traduction automatique neuronale

## Pourquoi les systèmes à règles sont toujours utiles ?



# Correction automatique et traduction automatique neuronale

## Pour les systèmes à règles sont toujours utiles ? Interprétabilité, interaction

MerciApp

Fonctionnalités Extension Avis\* Tarifs

Connexion

Créer un compte

### Mon document

Paul a défini un programme qui a retenu l'intention du jury. Il défendra son projet original et aura peut être l'opportunité de pouvoir accéder à la finale qui aura lieu mercredi 17 novembre 2020 au sain de nos locaux. Paul était d'ailleurs très content de nous avoir vu mardi dernier.

Verbe conjugué suspect ici

a... défini

a défini

Dans cette phrase, le mot « **défini** », qui correspond à une forme conjuguée du verbe « **définir** », est situé immédiatement après l'auxiliaire « **a** », ce qui est suspect.

715 000  
Formes de mot

1,5 million  
de combinaisons de mots

+3 000  
Règles sémantiques

+10 000  
Règles syntaxiques

300 000  
Sens des mots

+200  
Règles typographiques

200 000  
Nœuds dans notre ontologie

250 millions  
de mots dans notre corpus corrigé

[...]

# Correction automatique et traduction automatique neuronale

Les systèmes basés sur la traduction automatique sont utiles dans les cas où la performance est plus importante que l'interprétabilité

Par exemple ?

# Correction automatique et traduction automatique

Les systèmes basés sur la traduction automatique sont utiles dans les cas où la performance est plus importante que l'interprétabilité

- ❖ Extraction des variables utiles à l'évaluation automatique de productions écrites

# Evaluation automatique de productions écrites

- la plupart des systèmes d'AES utilisent systématiquement des variables basées sur les caractéristiques morphosyntaxiques des mots d'un texte
- nombre de types (si erreurs, nombre de types ↑, Crossley et al (2014))
- The sentence "A rose is a rose is a rose" contains three word types: three word tokens of the type a, three word tokens of the type rose, and two word tokens of the type is.

# Evaluation automatique de productions écrites

- la plupart des systèmes d'AES utilisent systématiquement des variables basées sur les caractéristiques morphosyntaxiques des mots d'un texte
- Distribution des parties du discours (pourcentage de noms, de verbes, d'adverbes...)
- Les analyseurs morphosyntaxiques en traitement automatique des langues (TAL) pour le français étant souvent entraînés sur des corpus en français standard peu bruité (le French Treebank (Abeillé et al., 2003))
- la performance de ces outils se trouve dégradée dès lors qu'il s'agit de traiter des textes de nature différente tels que des textes provenant du Web (Baranes & Sagot, 2014), des messages textos (Beaufort et al., 2010), ou encore des corpus oraux (Benzitoun et al., 2012).
- En ce qui concerne les corpus d'apprenants, Van Rooy & Schäfer (2003) constate que la précision de l'annotation des parties du discours (POS tagger) augmente de plus de 6% lorsque les erreurs orthographiques sont enlevées d'un corpus d'apprenants d'anglais. Sur 593 mots erronés, 49.2% reçoivent une annotation POS incorrecte.

# Evaluation automatique de productions écrites

- texte normalisé -> moins d'erreurs -> meilleures métriques lexicales + meilleure précision d'annotation syntaxique -> meilleur système de prédiction
- C'est une hypothèse à vérifier car peu d'études d'évaluation ☺

# Correction automatique et traduction automatique neuronale

- Quel est le lien ?
- traduction automatique < natural language generation (source - cible)
- text-to-text generation : résumé automatique, simplification de textes, correction d'erreurs
- ❖ correction d'erreurs (source = texte comprenant des erreurs, cible = texte sans erreurs)

# Faire comprendre la langue aux machines (réseau de neurones)

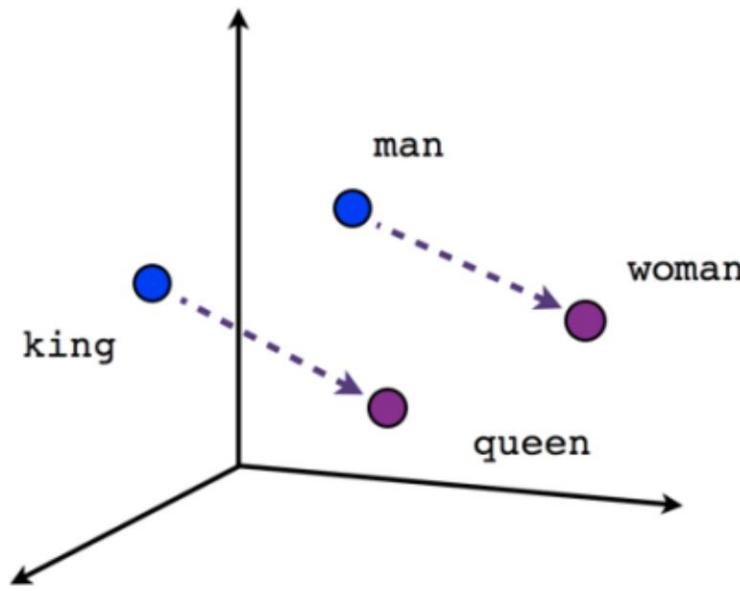
- La langue sous sa forme écrite est constituée d'unités discrètes

Pour l'ordinateur

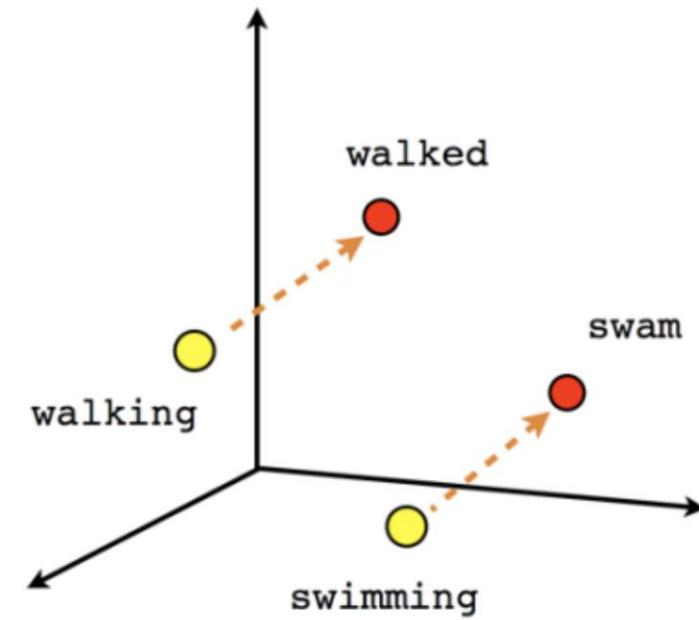
chat à char -> passage discret, il n'existe rien d'intermédiaire entre les deux unités

degré de noirceur pour les images (qch. entre chat et char)

# Pour une représentation continue



Male-Female



Verb tense

# Une représentation continue

---

- début : word2ve  
(Mikolov et al. 2013)

hypothèse distributionnelle

---

## Efficient Estimation of Word Representations in Vector Space

---

Tomas Mikolov  
Google Inc., Mountain View, CA  
tmikolov@google.com

Kai Chen  
Google Inc., Mountain View, CA  
kaichen@google.com

Greg Corrado  
Google Inc., Mountain View, CA  
gcorrado@google.com

Jeffrey Dean  
Google Inc., Mountain View, CA  
jeff@google.com

**‘You shall know a word by  
the company it keeps’**

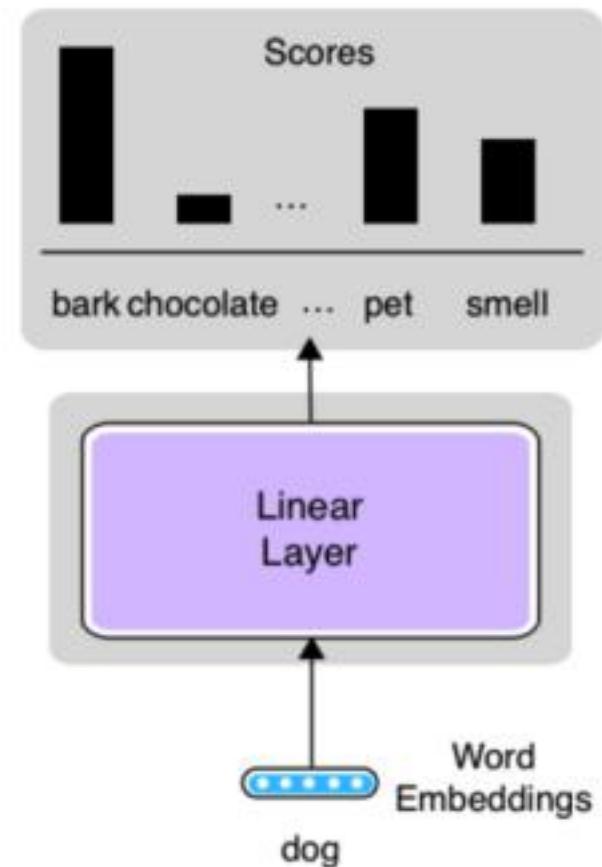
Firth, John R., 1957. *Modes of meaning*. Oxford: Oxford University Press.

# Implémentation (modèle Skip-Gram)

- Ce que l'on sait -> quels mots entourent le mot <dog> (distribution de probabilité)

- Tâche :

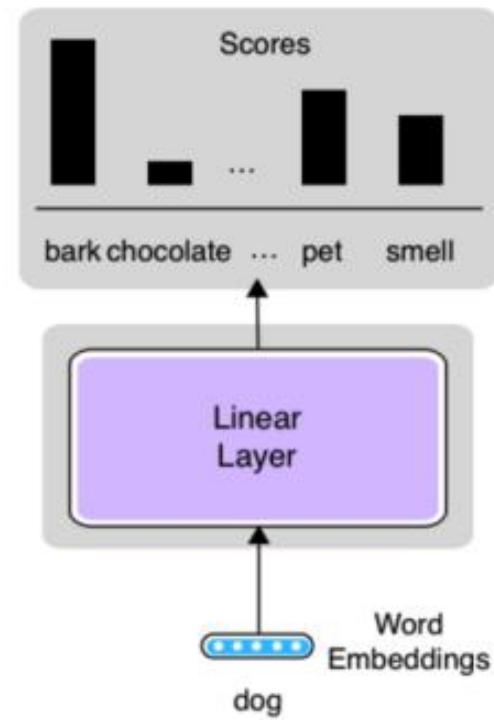
Etant donné un mot,  
prédirer la probabilité des mots environnants



# Implémentation

- Supposons qu'on représente les mots <heard, a, dog, barking, in> par  $[x_i, y_i, Z_i]$ , comment fait-t-on pour arriver à la distribution de [0.1,0.2,0.3,0.4] ?
- $(x_i, y_i, Z_i) \rightarrow$  word embeddings

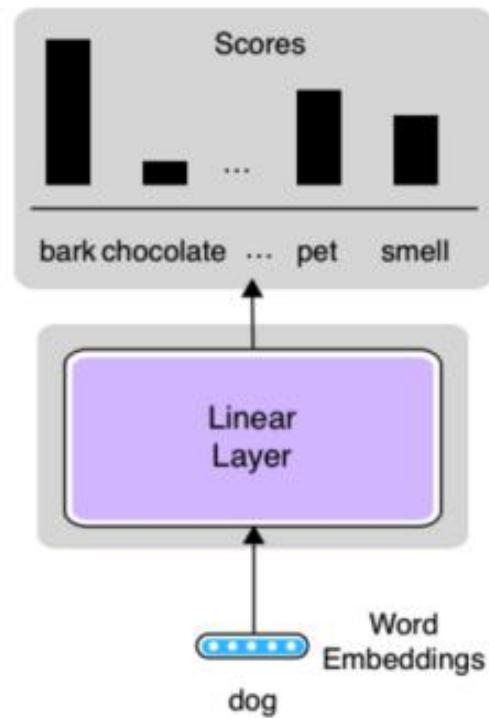
```
● ● ●  
def linear2_to_1(x1, x2):  
    return w1 * x1 + w2 * x2 + b
```



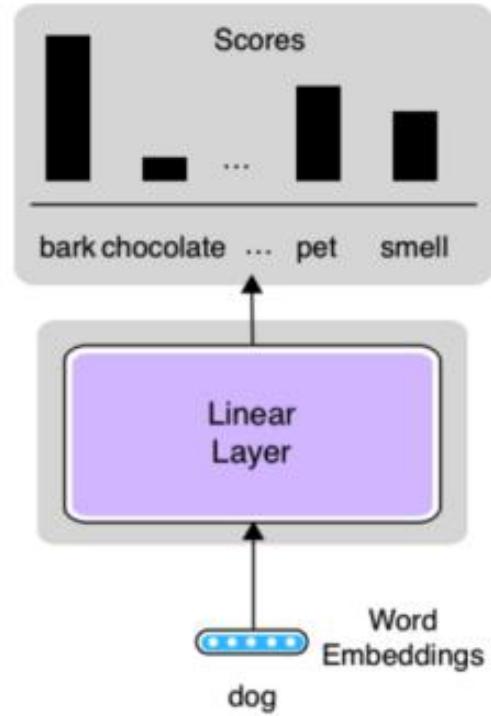
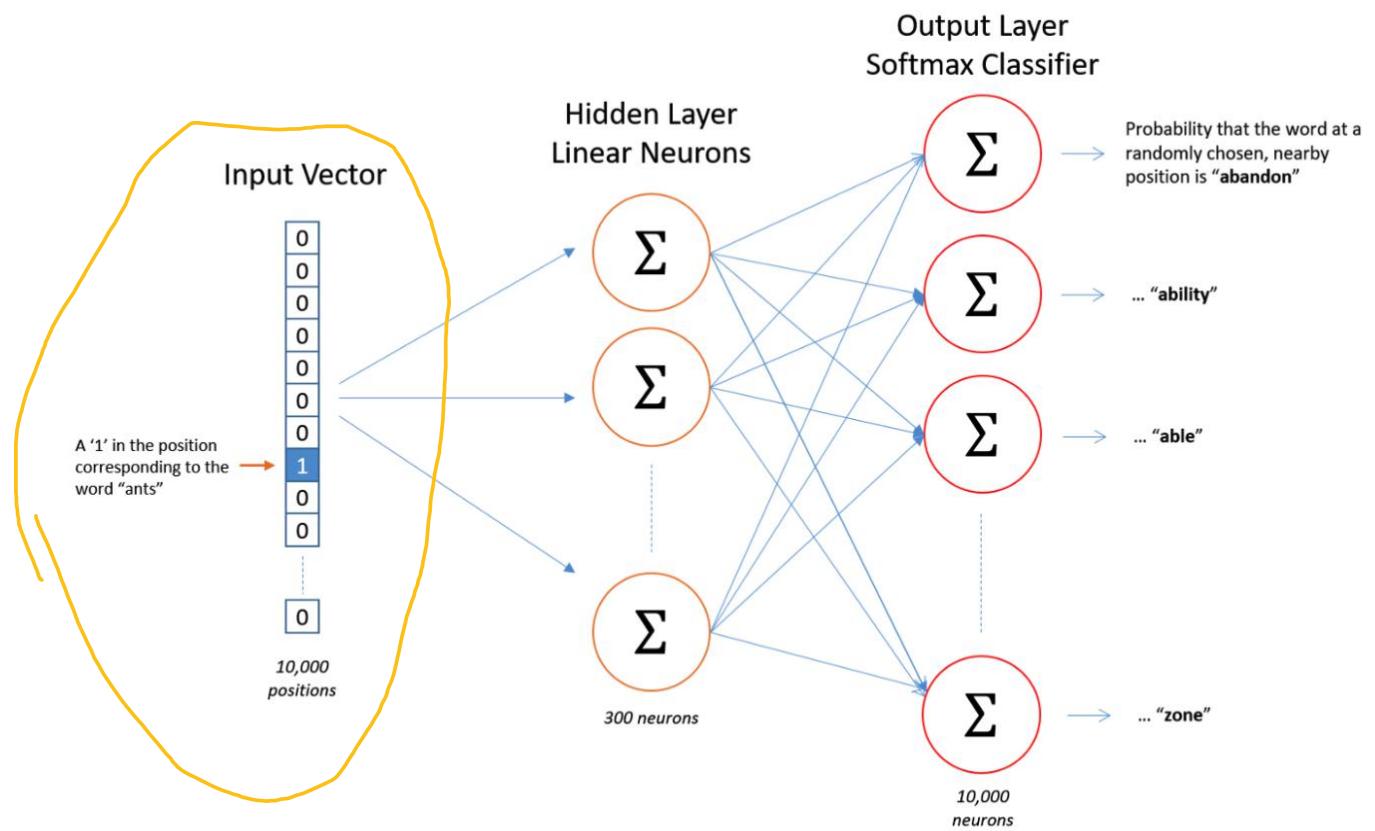
# Implémentation

- Supposons qu'on représente les mots <heard, a, dog, barking, in> par  $[x_i, y_i, Z_i]$ , comment fait-t-on pour arriver à la distribution de [0.1,0.2,0.3,0.4] ?
- $(x_i, y_i, Z_i) \rightarrow$  word embeddings

```
● ● ●  
def linear3_to_4(x1, x2, x3):  
  
    y1 = w11 * x1 + w12 * x2 + w13 * x3 + b  
    y2 = w21 * x1 + w22 * x2 + w23 * x3 + b  
    y3 = w31 * x1 + w32 * x2 + w33 * x3 + b  
    y4 = w41 * x1 + w42 * x2 + w43 * x3 + b  
    return (y1,y2,y3,y4)
```



# Implémentation



# Comment représenter une phrase ?

- Faire la moyenne des word embeddings (ordre...)

Jean adore le chat. = Le chat adore Jean.

# Comment représenter une phrase ?

- RNN

Actualiser la représentation mentale d'une phrase au fil des mots.

Jean adore le chat.

```
1 state = init_state()
2 state = update(state, v("Jean"))
3 state = update(state, v("adore"))
4 state = update(state, v("le"))
5 state = update(state, v("chat"))
6 state = update(state, v("."))
```

# Améliorations

- On comprend mieux les mots quand on lit la phrase dans les deux sens.
- Jean lui montre le chat. (lui est souvent suivi d'un verbe et le précédent d'un verbe)

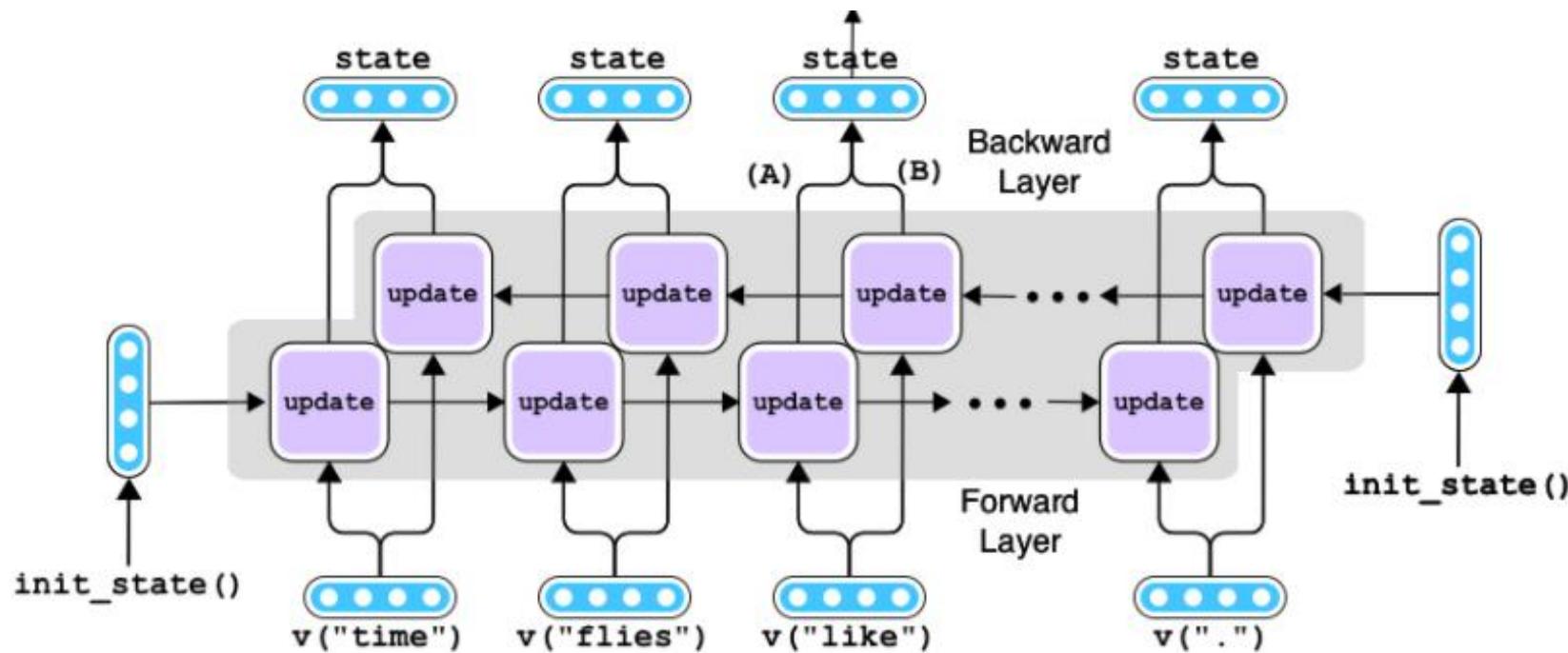


montre !=



# RNN bidirectionnel

- mettre bout à bout deux embeddings (forward et backward)



# Problème

- Plus la phrase est longue, plus il est difficile d'encoder la phrase (se rappeler des mots des deux extrémités)
- Traduction automatique...

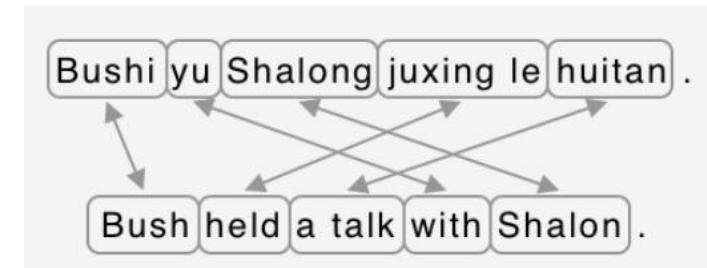
# Petite histoire de machine translation

- système expert, Georgetown-IBM experiment, seconde guerre mondiale
- trouver la traduction d'un mot dans un dictionnaire de correspondance
- rédiger des règles pour aligner les mots
- Vite intenable !

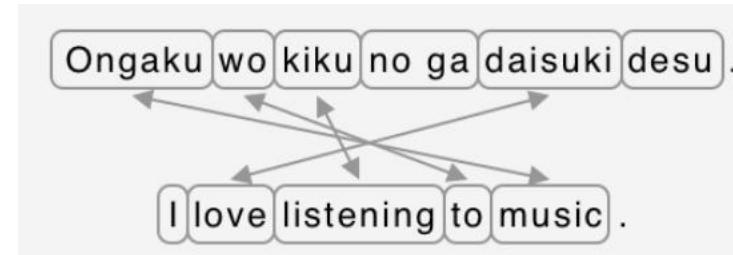
# spanish



# chinese



# japanese



# 1980s

- Statistical machine translation (SMT), plus de système expert
- utiliser un corpus bilingue (bitextes)
- trouver des traductions candidates susceptibles de correspondre à la phrase originale + modèle de langue pour choisir la bonne candidate
- modèle de langue (prédire la probabilité d'une phrase)
- $\text{Im}(\text{elle est belle}) > \text{Im}(\text{elle est beau})$

# Moses



**MOSES**  
statistical  
machine translation  
system

## 1. Moses

Overview

Manual

Online Demos

FAQ

Mailing Lists

Get Involved

Recent Changes

## 2. Getting Started

Source Installation

Baseline System

Packages

Releases

Sample Data

Links to Corpora

## 3. Tutorials

[Main](#) » [HomePage](#)

## Welcome to Moses!

Moses is a **statistical machine translation system** that allows you to automatically train translation models for any language pair. All you need is a collection of parallel text pairs, and the algorithm quickly finds the highest probability translation among the exponential number of choices.

### News

- **5 October 2017** Moses v 4.0 has been [released!](#)
- **8 September 2016** [Moses2](#), a fast drop-in replacement for the Moses decoder
- **12 December 2015** [Add a new feature function to Moses](#)
- **17 June 2015** [Slate](#) for Windows
- **15 June 2015** Moses, and more, on Amazon cloud [Box](#)
- **1 June 2015** Developing Moses with Eclipse [video](#)
- **4 February 2015** Moses v 3.0 has been [released!](#)
- **21 July 2014** Moses now has nightly [speed tests](#)
- **14 July 2014** [How to compile Moses with Eclipse](#)
- **4 March 2014** Bug fix release for Moses, now version 2.1.1
- **3 February** The [2014 Machine Translation Marathon](#) will take place in Trento, Italy from 8-13th September.
- **21 January 2014** Moses v 2.1 has been [released!](#)
- **26 March 2013** The [2013 Machine Translation Marathon](#) (MTM2013) will take place in Prague, Czech Republic from 9-14th September
- **5 March 2013** What do you want to see in Moses v2.0? See [here](#) for projects and how to suggest them.
- **28 January 2013** Moses v 1.0 has been [released!](#)
- **12 October 2012** Moses v 0.91 released
- **February 2012:** Moses development is being supported by the EU under the **MosesCore** project
- **September 2011:** Moses now has a [cruise control page](#) to see the status of the current builds
- **September 2011:** Moses is now hosted on [github](#)



[Main](#) » [HomePage](#)

## Welcome to Moses!

Moses is a **statistical machine translation system** that allows you to automatically train translation models for any language pair. All you need is a collection of parallel text pairs, and the algorithm quickly finds the highest probability translation among the exponential number of choices.

from 8-13th September.

e in Prague, Czech Republic from 9-14th September to suggest them.

re project  
ent builds

# En 2015

- Neural machine translation (NMT)

réseau de neurones (RNN bidirectionnel)

# Avantage de NMT

- «Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance » (Bahdanau et al. 2016)

De manière classique, SMT a besoin d'un modèle de langue pour choisir la bonne traduction, approche composée

Composante 1 -> trouver des bouts de traduction, composante 2 -> réagencement des segments 3 -> modèle de

# Avantage de NMT

- En NMT, le système est structuré de manière à entraîner les composantes en même temps

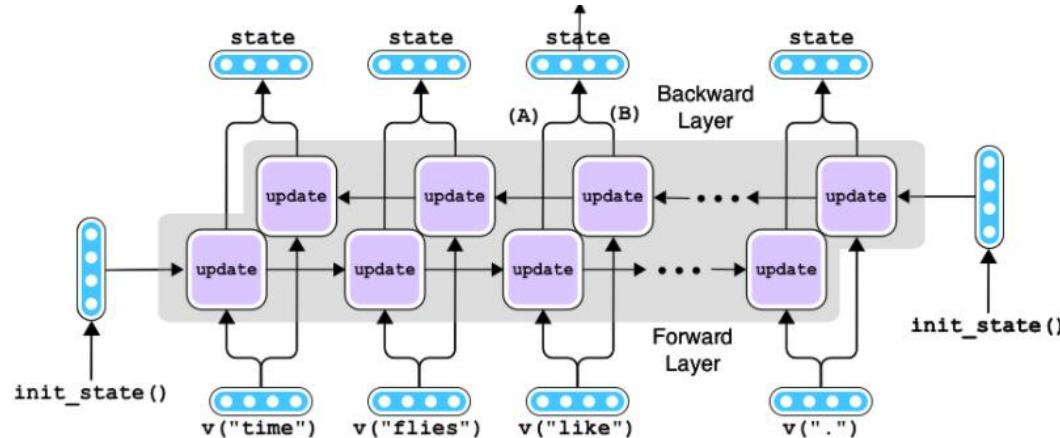
## Mécanisme d'attention

«We conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder–decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly.»

(Bahdanau et al. 2016)

# Retournons au RNN

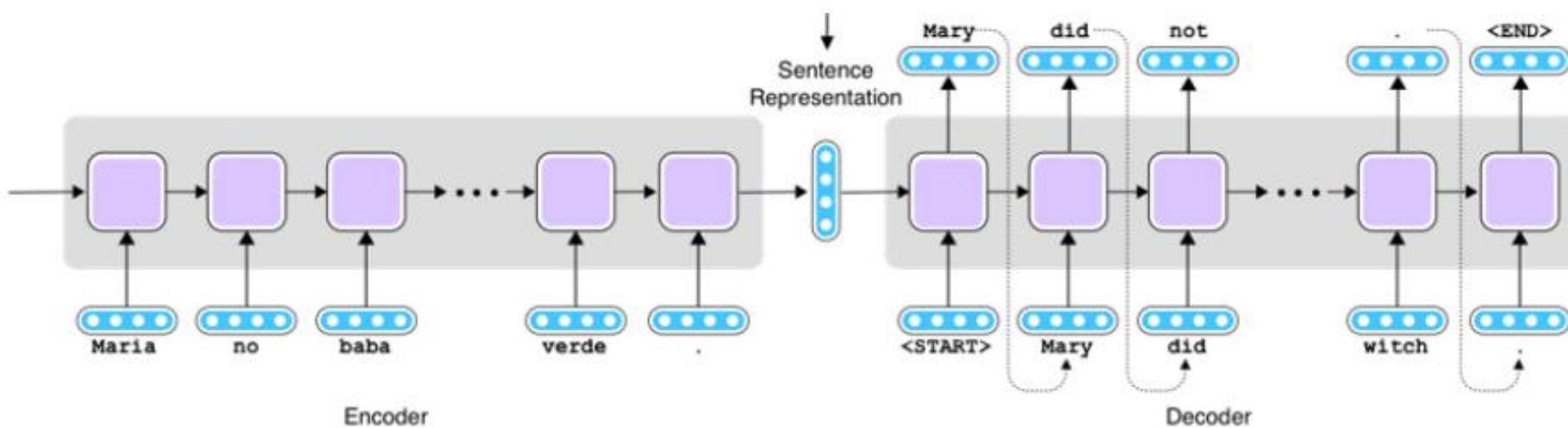
- Une phrase, quelle que soit sa longueur, est encodée dans un seul embedding [1.2,3.4,...,2.3]



The usual RNN, described in Eq. (1), reads an input sequence  $\mathbf{x}$  in order starting from the first symbol  $x_1$  to the last one  $x_{T_x}$ . However, in the proposed scheme, we would like the annotation of each word to summarize not only the preceding words, but also the following words. Hence, we propose to use a bidirectional RNN (BiRNN, Schuster and Paliwal, 1997), which has been successfully used recently in speech recognition (see, e.g., Graves *et al.*, 2013).

# Retournons au RNN

Le décodeur (traducteur) ne dispose d'un seul embedding pour traduire toute la phrase, quelle que soit sa longueur.



# Mécanisme d'attention (Bahdanau et al. 2016)

- Que fait un traducteur humain ?

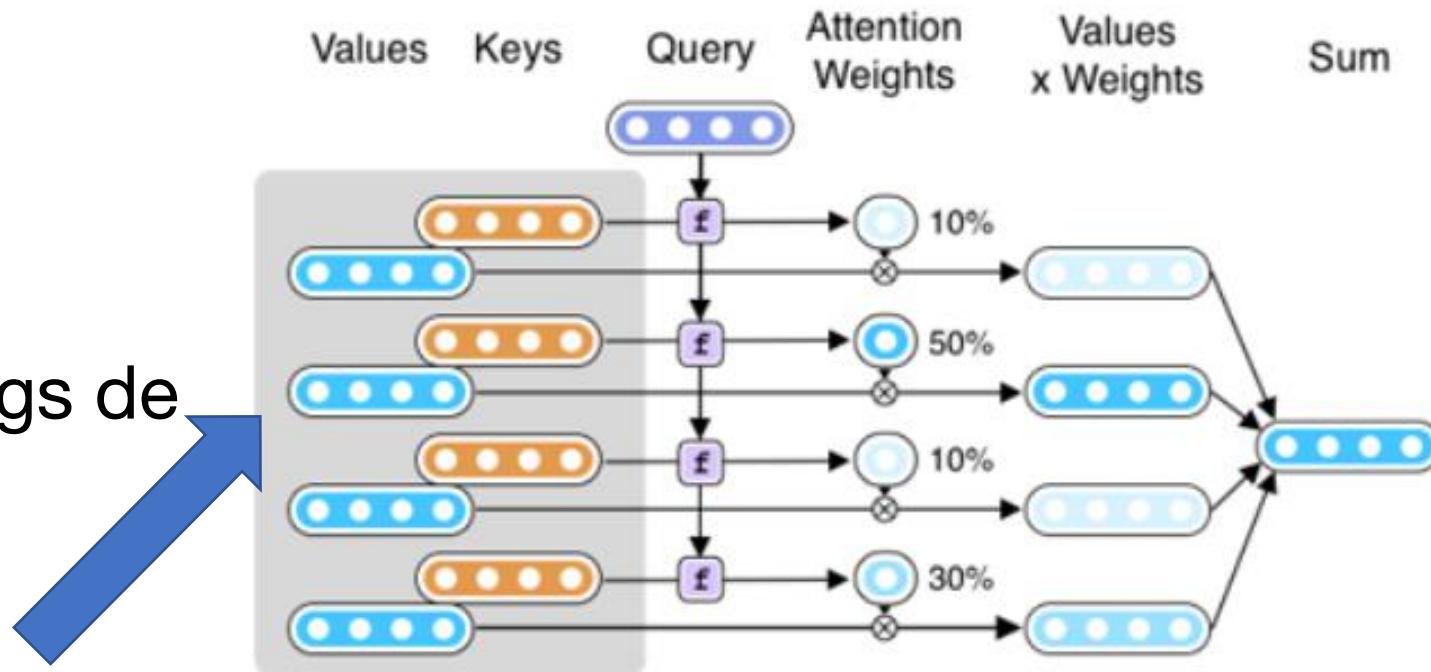
Un traducteur humain ne mémorise pas la phrase, il traduit petit à petit, en se référant constamment à la partie pertinente de la phrase d'origine.

Au fil de la traduction, il focalise son attention de manière dynamique sur une partie de la phrase.

# Implémenter l'attention

Eléments principaux :

1, keys = values =  
les inputs (word embeddings de  
chaque mot)

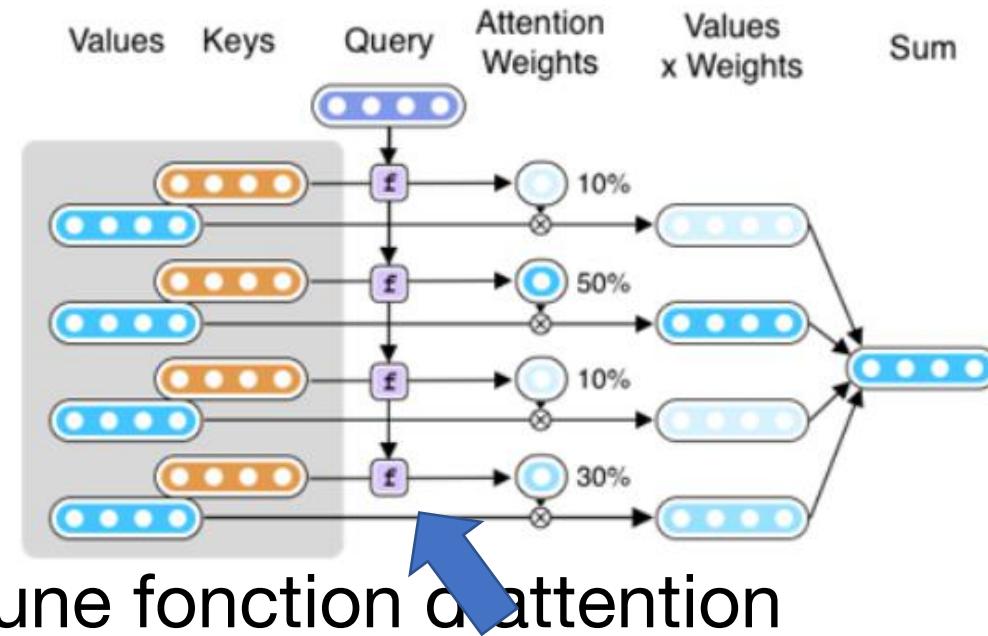


# Implémenter l'attention

Eléments principaux :

1, keys = values =  
les inputs (embeddings des mots)

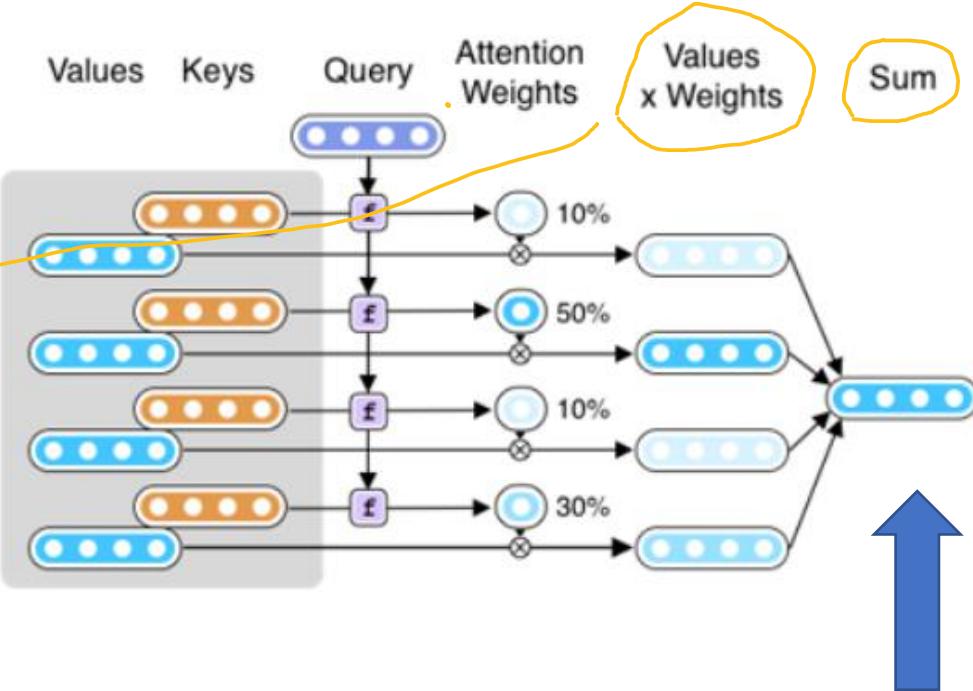
2, Chaque key est comparée avec  
une query (le mot à traduire) grâce à une fonction d'attention  
 $f$  -----> une distribution de weights pour chaque  
input. Plus le poids est élevé, plus cet input est pertinent.



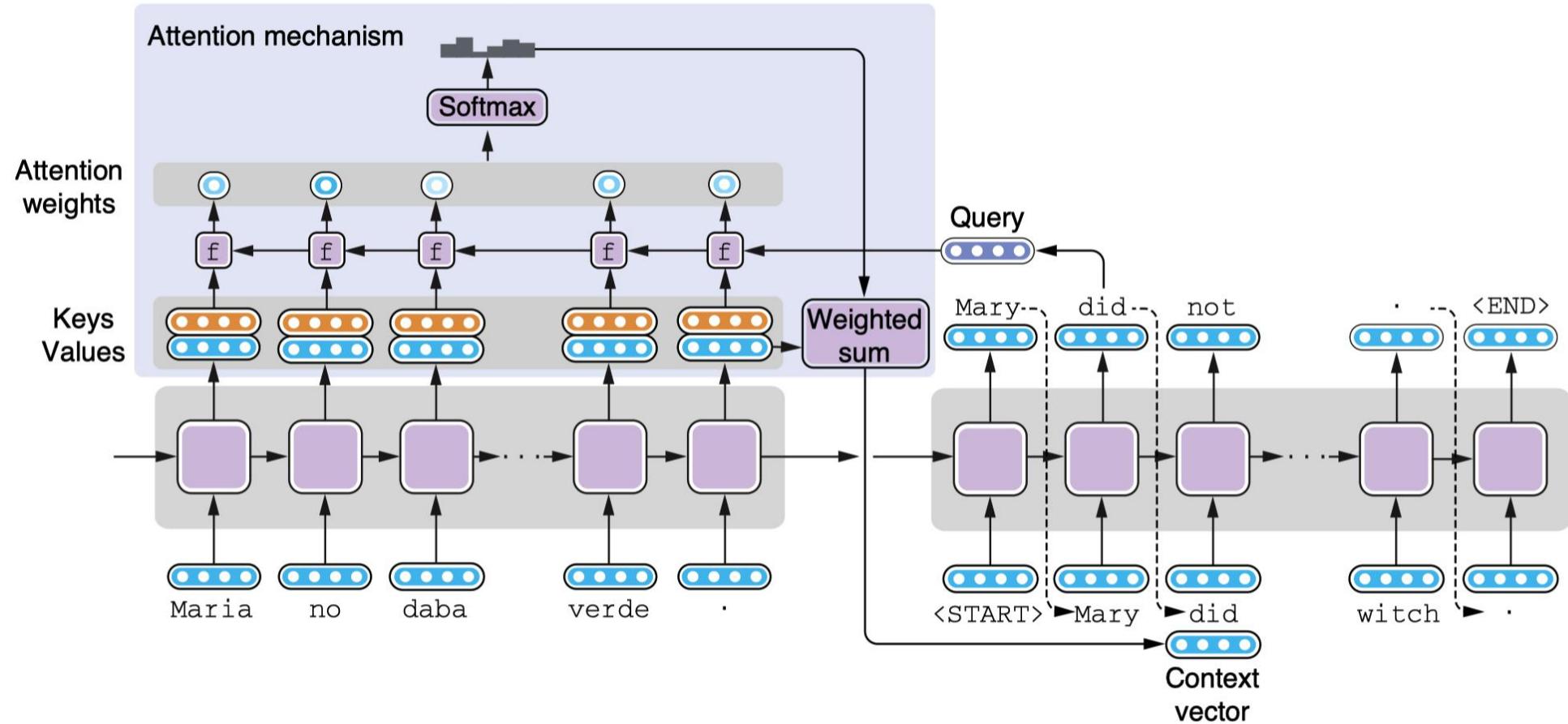
# Implémenter l'attention

Eléments principaux :

3, Les values sont pondérés avec leur poids respectif pour produire une somme



# Mécanisme d'attention



# Avantage de NMT

- En NMT, le système est structuré de manière à entraîner les composantes en même temps -> end-to-end

Plus besoin d'utiliser un modèle de langue ni un modèle d'alignement

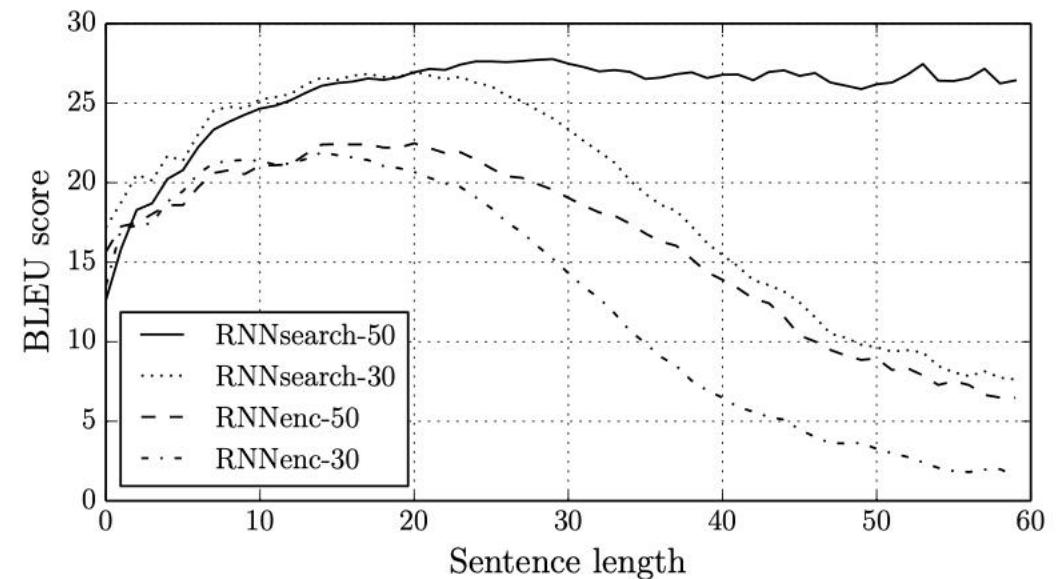
We parametrize the alignment model  $a$  as a feedforward neural network which is jointly trained with all the other components of the proposed system.

# Résultats

- RNNsearch = RNN avec attention, RNNenc = RNN sans attention
- WMT '14 (corpus parallèle français-anglais)
- corpus : 348M mots | test : 3003 phrases

30, 50 = longueur de phrases maximale utilisée pendant l'entraînement

(Bahdanau et al. 2016)



- «One of the motivations behind the proposed approach was the use of a fixed-length context vector in the basic encoder-decoder approach. We conjectured that this limitation may make the basic encoder-decoder approach to underperform with long sentences.»
- RNNsearch50, especially, shows no performance deterioration even with sentences of length 50 or more.
- RNNsearch-30 even outperforms RNNencdec-50.

# Self-attention (Vaswani et al. 2017)

- The Law will never be perfect, but its application should be just.

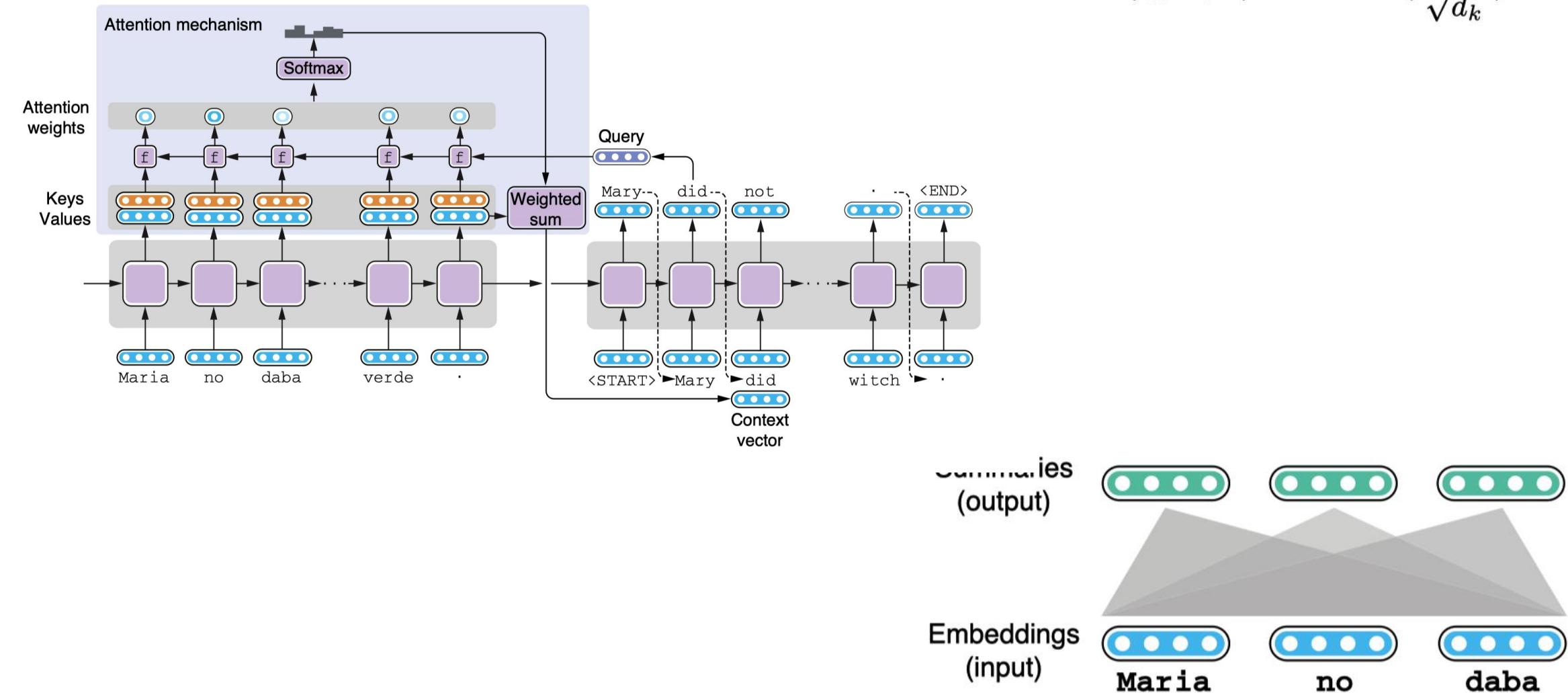
résolution d'anaphore

Avec RNN, le réseau de neurones a besoin de mémoriser le mot “The Law” jusqu’à l’apparition du mot its.

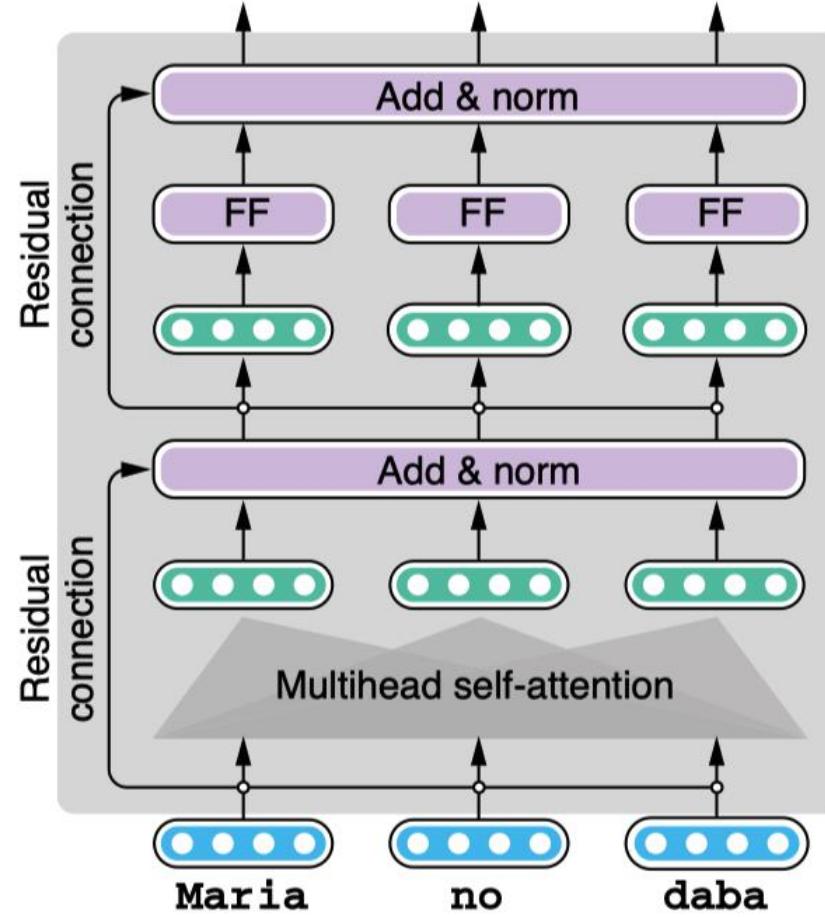
Simple rules like “replace it with the nearest noun that appeared before” will suffice. Self-attention is better at learning such long-range dependencies, as we’ll see later.

# Cross-attention

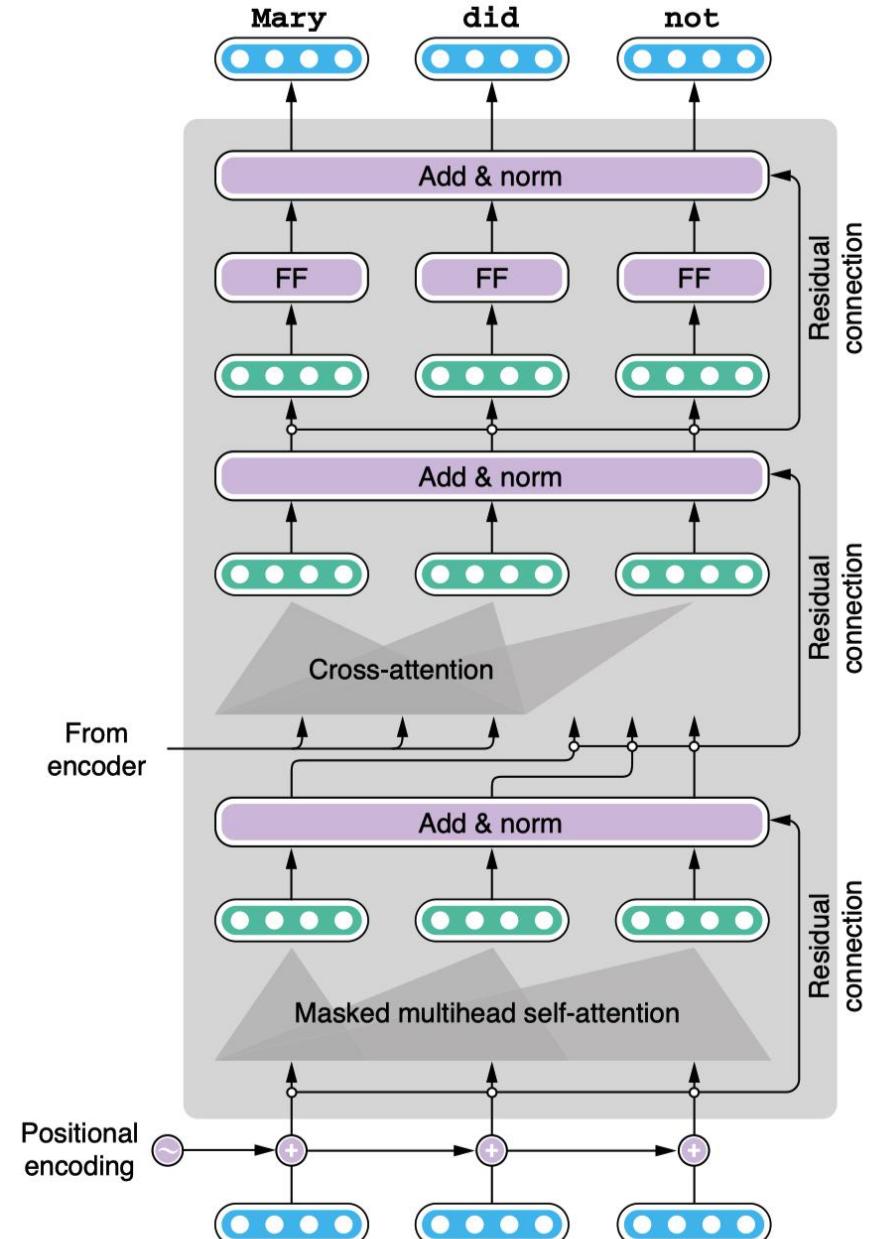
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



# Self-attention dans l'architecture Transformer



Vaswani et al. 2017



# Teaser

- tisimptant too spill chck ths dcment.  
❖ It's important to spell check this document.
- The book Tom and Jerry put on the yellow desk yesterday wer about NLP.  
❖ The book Tom and Jerry put on the yellow desk yesterday was about NLP.
- The books Tom and Jerry put on the yellow desk yesterday wer about NLP.  
❖ The books Tom and Jerry put on the yellow desk yesterday were about NLP.

# Grammaly

Untitled document

tisimptant too spll chck ths dcment.

## 5 All suggestions

### • SPELLING

tisimptant → visitant

The word **tisimptant** is not in our dictionary. If you're sure this spelling is correct, you can add it to your personal dictionary to prevent future alerts.

□+ Add to dictionary



• spll · Correct your spelling

• chck · Correct your spelling

• ths · Correct your spelling

• dcment · Correct your spelling

# Grammaly

Untitled document

The book Tom and Jerry put on the yellow desk yesterday **wer** about NLP.

## 1 All suggestions

### • SPELLING

**wer** → **were**

The word **wer** is not in our dictionary. If you're sure this spelling is correct, you can add it to your personal dictionary to prevent future alerts.

 Add to dictionary



# MerciApp



salut julia,je vous preparer un bon plat, des poisson a la souce d'asie , il y aussi des aceser de fruits de saison, et des fruits exutique, et des jus de fruit fait maison, en va faire un bon soiree .

# Perspectives pour le français

- Aucun correcteur grammatical utilisant l'approche NMT
- Peu de corpus parallèles pour entraîner des systèmes de correction NMT (Granger, 2003)
- Peu de travaux évaluant la contribution de la normalisation (correction d'erreurs dans notre cas) à AES
- Le NLP est passionnant ☺