Introduction à l'apprentissage profond en recherche d'Information

-

Deep Learning and Information Retrieval

Plan

- Retour sur la représentation des textes
  - représentation distribuée de mots, *word embedding*
- Brève introduction à l'apprentissage profond (Deep learning)
- Modèles neuronaux de RI (Neural IR Models)
- Modèles de transformeurs (*Transformers)*

Retour sur la représentation des textes :

Apprentissage de représentation de mots (textes)

Représentation vectorielle

D : The noble Brutus was ambitious

- Bag of words, ngrams
  - Words as atomic units (noble, Brutus, ambigous)
- Word representation
  - (w_1, w_2 …., $w_{|V|}$), the whole vocabulary a million (or a billion) of words
  - Term vector :
    - term vector : (0,0,0,0,0,0,0.,0,..,1, …, 0) (appelé **One hot vector**)
  - Text (document) is represented
    - (0,1,0,0,1,0,0.,1,..1, …, 0) (presence/absence of words
    - (0,2,0,0,3,0,0…,1,..5, …, 0) (term frequency)
    - (0, 0.25, 0, 0, 0.29, 0, 0,…, 0.02, …, 0.6,…) : Weighted terms (tf*idf)
  - Document collection is represented as a matrix

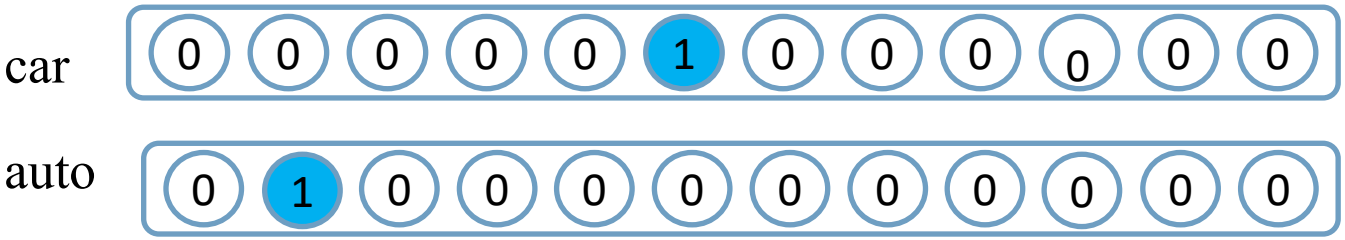
	d_0	d_1	d_2	…	d_j	…	$d_{\|D\|}$
t_0	1	0	0	0	2		1
t_1	0	0	0	0	1		1
t_2	1	1					
…							
…							
$t_{\|V\|}$	1	0	0				1

Représentation vectorielle

- More sophisticated word weighting
  - Information Theory
    - [Amati et al 2004], [S. Clinchant & Gaussier, 2010]
  - Probabilistic Model
    - BM25 [Robertson et al. 1994]
    - Statistical Language model [Ponte & Croft 1998], [ChengXiang et al 2002]
      - Assigns a probability to a sequence of words
      - p("Today is Wednesday") ≈ 0.001
      - p("Today Wednesday is") ≈ 0.0000000000001

Limites de la representation par sac de mots

- Limits of bag of words (One Hot vector)
  - Synonymy, different word with the same meaning “Car” and “auto”,
  - Polysemy, same word but different meaning “salsa (dance)”, “salsa (sauce)”
  - Example :



- Beyond bag of words → Semantic representation (represent the meaning of a word)
  - Use thesaurus (WordNet) : containing a list of synonym sets and hypernyms
  - Distributional similarity based representations

Vers une representation sémantique des mots

- Distributional similarity based representations
  - ”You shall know a word by the company it keeps”

Firth, J. R. (1957).

  - Word is represented according to its context (its neighbors)

…government debt problems turning into	**banking**	*crises as happened in 2009…*
…saying that Europe needs unified	**banking**	*regulation to replace the hodgepodge…*
…India has just given its	**banking**	*system a shot in the arm…*

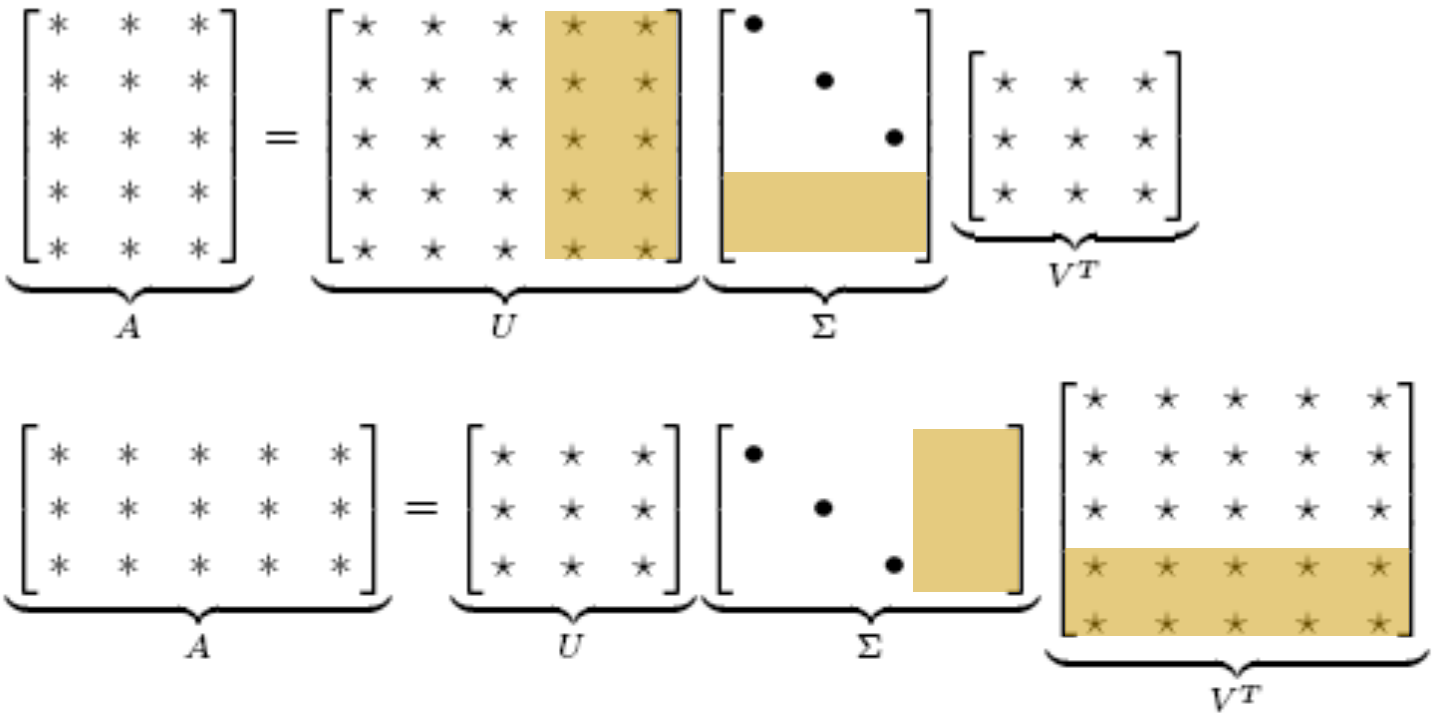
  - How to make neighbors represent words?
    - Represent a word in a dense (low dimensional) vector (25 - 1000) built (learnt)
    - LSA (LSI)based on Matrix Factorization (SVD) [S. Deerwester et al 1988]
    - Neural Word Embedding: Word2Vec (CBOW, Ski-gram) [T. Mikolov et al 2013]

“car” : [0.015 , 0.001, -0.020, 0.015, 0.001]

Représentation continue (distribuée) des mots → Vers une representation sémantique des mots

Indexation sémantique latente (Latent Sematic Indexing)

- Illustration de la SVD



Indexation sémantique latente (Latent Sematic indexing)

- SVD (Singular Value Decomposition) $A=TSD^T$

Select k (300) top singular values

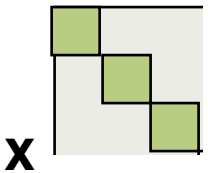
	d_0	d_1	d_2	...	d_j	...	$d_{\|D\|}$
t_0	1	0	0	0	2		1
t_1	0	0	0	0	1		1
t_2	1	1					
...							
...							
$t_{\|V\|}$	1	0	0				1

=

T

	c_0	c_1	c_2
t_0	...	-.	..
t_1	...	..	..
t_2	...	..	
...			
...			
$t_{\|V\|}$	..	..	..

S



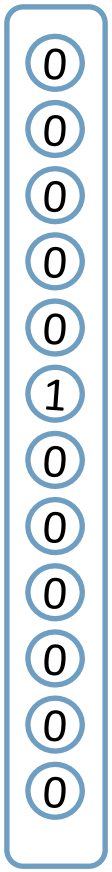
D^T

- La matrice D représente les documents dans le nouvel espace vectoriel (espace des concepts)
- La matrice T: la représentation sémantique d'un terme
- La fonction qui permet le passage des termes vers les concepts est $M= T[v,k].S^{-1}[k,k]$

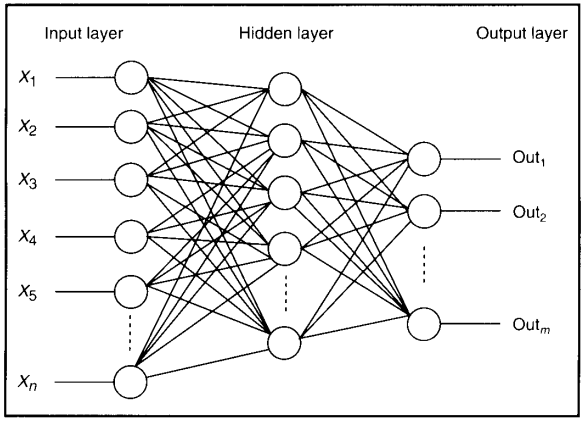
Approche neuronale (Neural Embedding)

- Utiliser des réseaux de neurones pour apprendre des représentations continues (denses de mots)

car



|V|=10^6

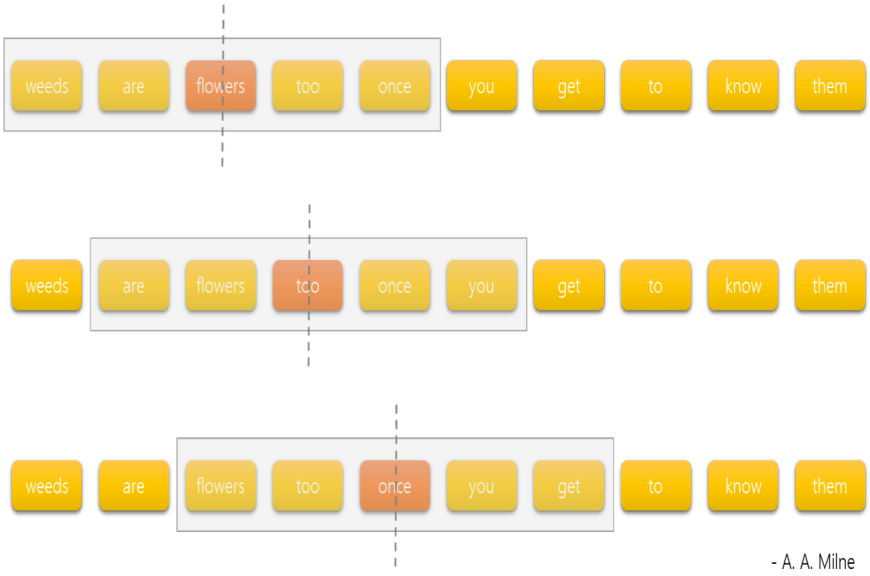


$$\begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

|V_dense|=100

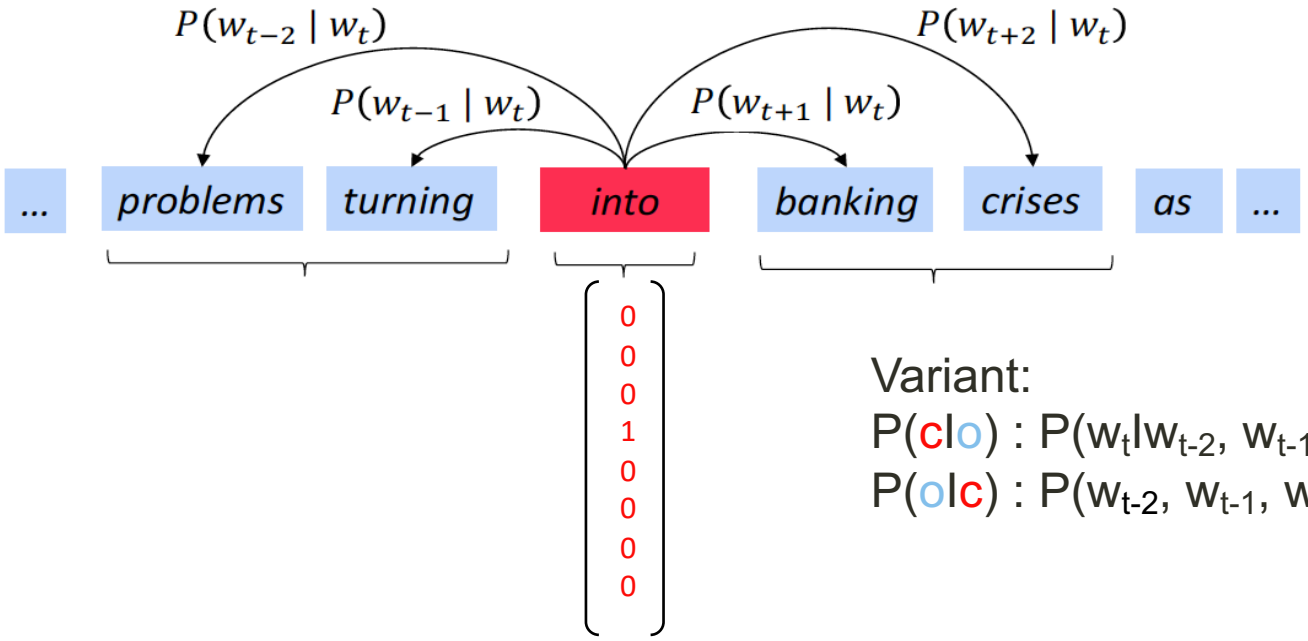
Word2vec

- Goal: simple (shallow) neural model learning from a large corpus of text (billion words)
- Every word in a fixed vocabulary is represented by a vector
- Go through each position t in the text, which has a center word c and context (“outside”) words o
- Predict center (c) word from its context(o) (Po/c) or vice-versa (P(c/o)



Word2Vec

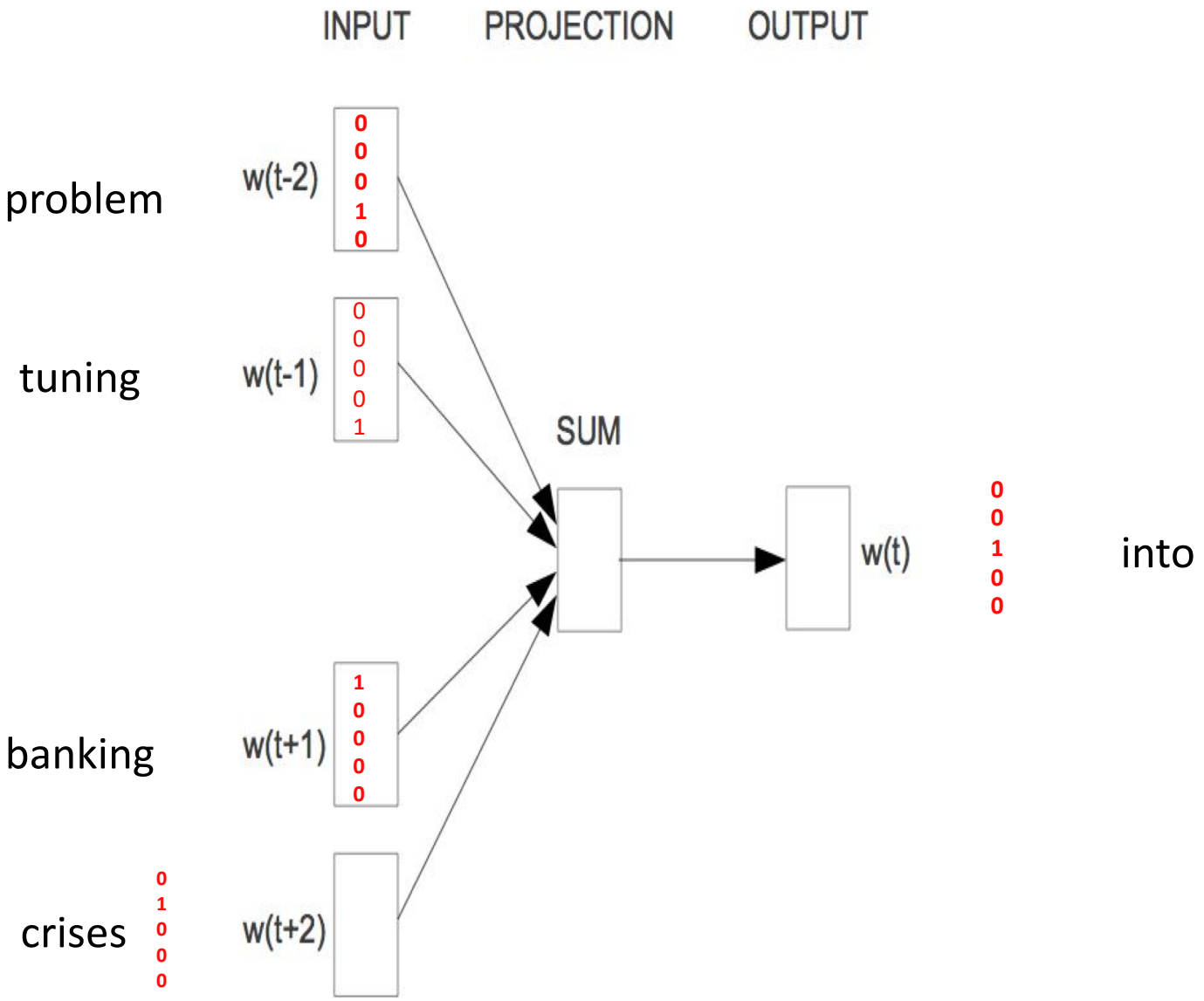
- Word2vec (Mikolov et al. 2013), the main Idea:
  - Go through each position in the text, which has a **center** word w_t and **context** ("outside") words w_{t-1}
  - Use the similarity of the word vectors for c and o to calculate the probability P(o|c) or P(c|o)



Variant:
P(c|o) : $P(w_t|w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2})$
P(o|c) : $P(w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}|w_t)$

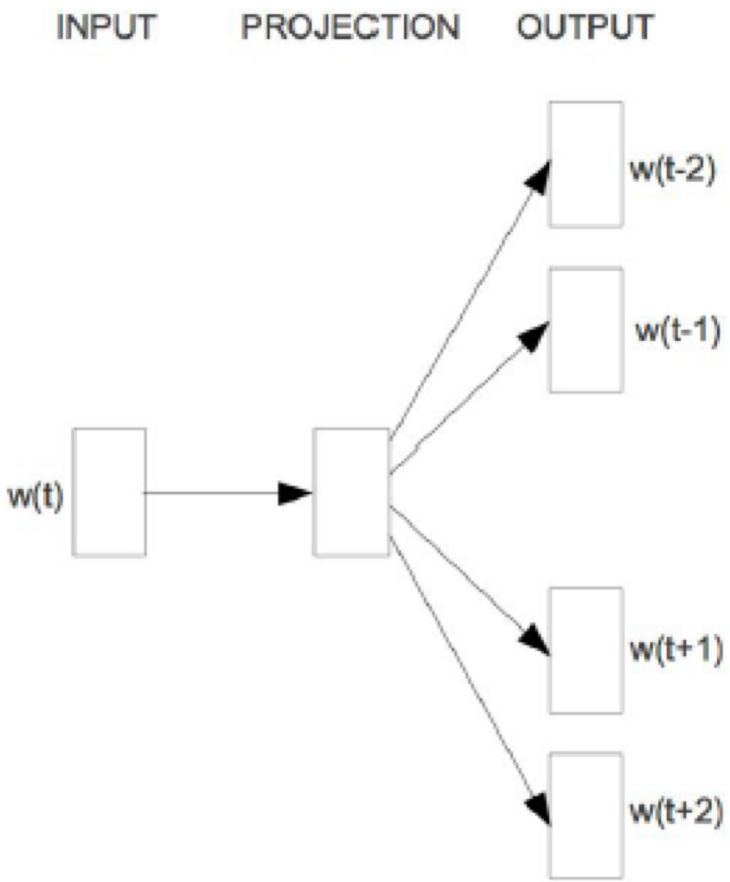
Word2Vec

- CBOW



Word2Vec

- Skip-Gram



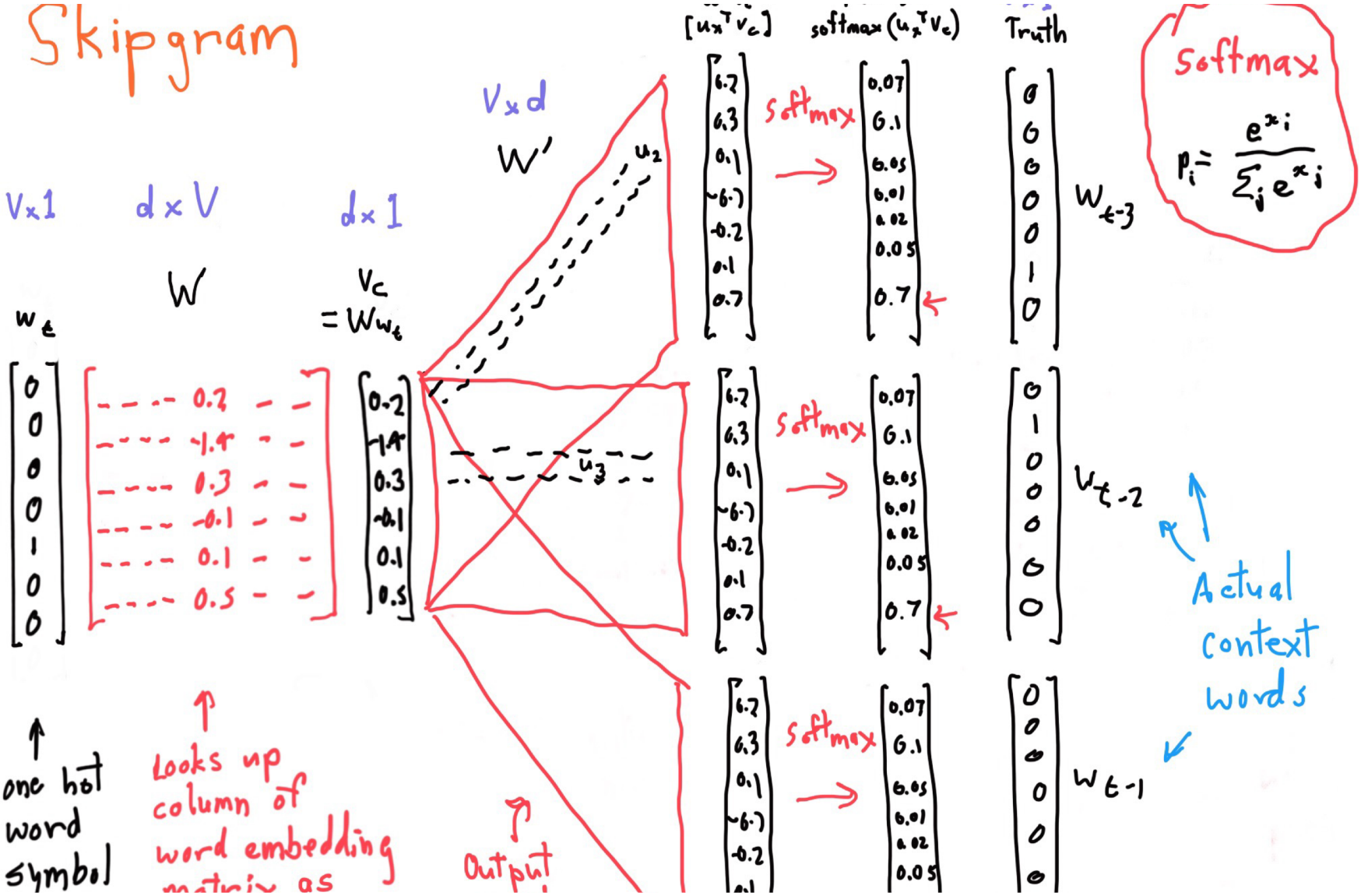
into

problem

tuning

banking

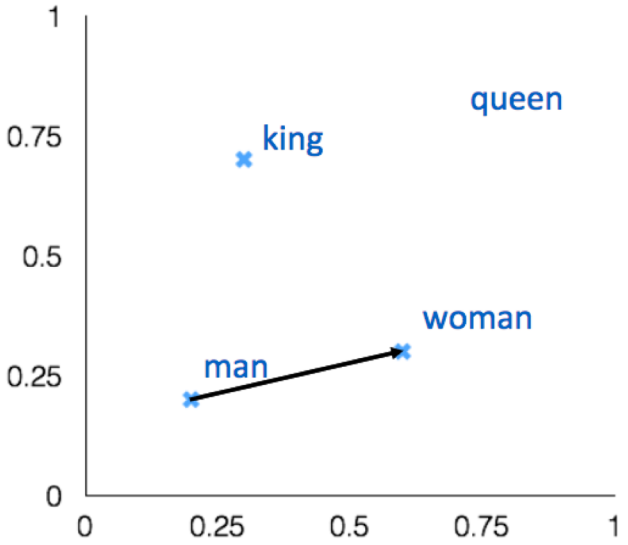
crises



Word2Vec

man:woman :: king:?

+	king	[0.30 0.70]
-	man	[0.20 0.20]
+	woman	[0.60 0.30]
	queen	[0.70 0.80]



Test word analogies [Mikolov 2014]

Word2Vec

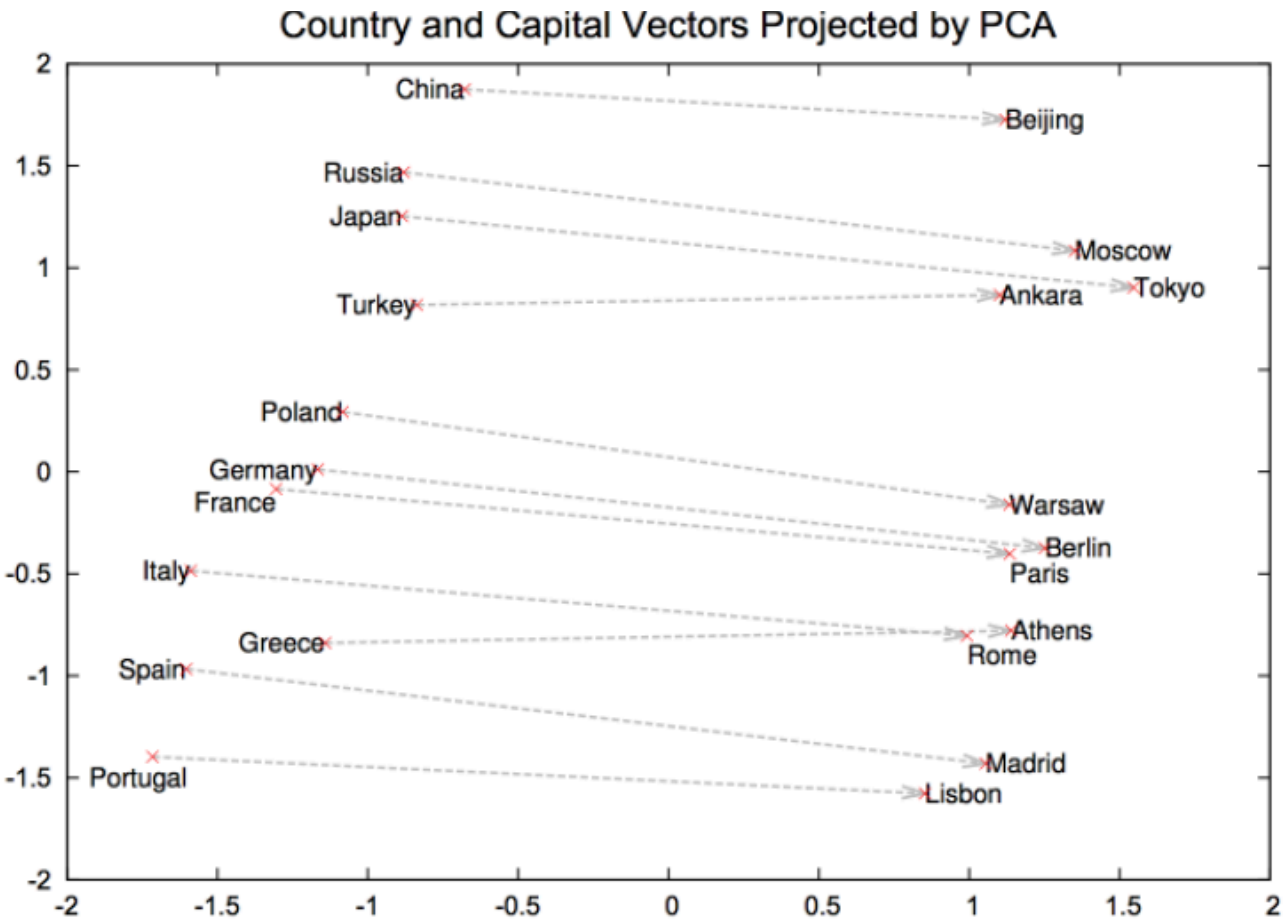


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.

Plan

- Rappel de quelques definitions sur l'apprentissage profond
- Deep learning et RI
  - Retour sur la representation des documents
  - Utilisation des *Word Embedding* en RI
    - Un peu de pratique Python

Utilisation de Word2Vec

- Un peu de pratique en Python
  - On peut construire son propre Word2Vec (operation fastidieuse) (il faut beaucoup, beaucoup de textes)
  - Utiliser des Word2Vec pré-entrainés (Google, Glove, …)

- Une bonne majorité des modèles de représentation de textes utilise la représentation distribuée (continue) des termes.
- On peut apprendre différents types de représentations: termes, sous-termes (n-grammes), caractères, etc.

- Beaucoup de modèles de RI exploitent ce type de représentations

Approche générale

- Extend traditional IR models
  - Term weighting, language model smoothing, translation of vocab
- Expand query using embeddings (followed by non-neural IR)
  - Add words similar to the query
- IR models that work in the embedding space
  - Centroid distance, word mover's distance

Pre-trained W2V

Query text

Document text

Generate query representation

Generate document representation

land, mascot, site
0.01 -0.013 0.02
0.03 0.25 0.03
0.105 0.15 0.4

Query vector

Doc. vectors

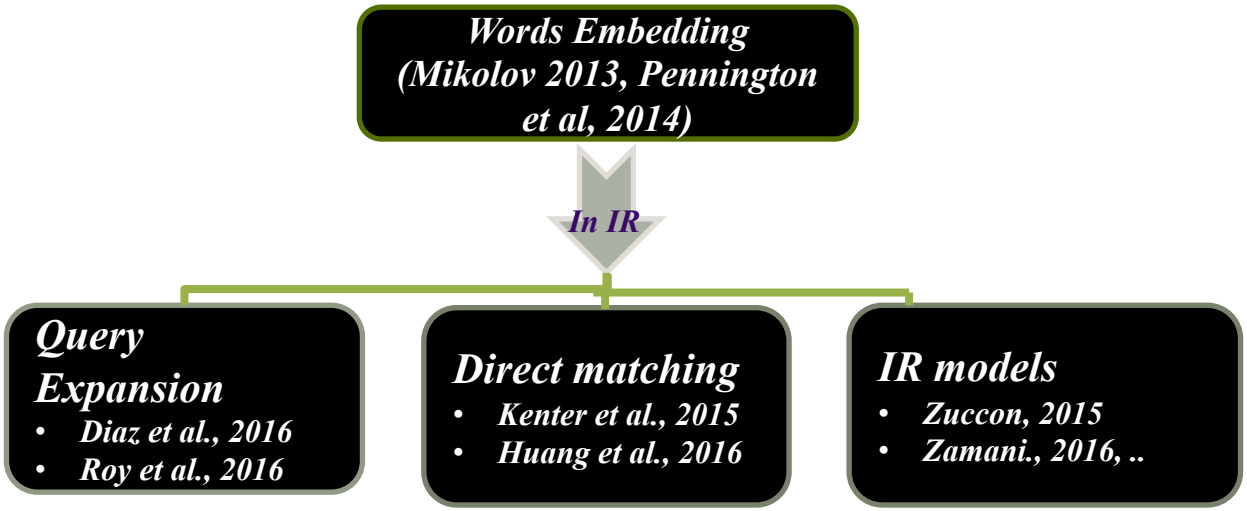
aryugu , asteroid, best,
-0,25 0,031 0.123 …
0,23 -0125 -0.54 …
0,15 0,135 -0.052 …

Agregation (sum, avg, …)

Agregation (sum, avg, …)

Estiate relevance

Trois classes d'approches



$$rsv(q,d) = \sum_{t \in q} \sum_{t' \in V} P(t,t').P(t'/d)$$

Task	Studies
Ad-hoc Retrieval	ALMasri et al. (2016), Amer et al. (2016), BWESG (Vulic and Moens (2015)), Clinchant and Perronnin (2013), Diaz et al. (2016), GLM (Ganguly et al. (2015)), Mitra et al. (2016), Nalisnick et al. (2016), NLTM (Zuccon et al. (2015)),Rekabsaz et al. (2016), Roy et al. (2016), Zamani and Croft (2016a), Zamani and Croft (2016b), Zheng and Callan (2015)

- Dual Embedding Space Model (Mitra et al 2016)
  - A document is represented by a centroid of its word vectors

$$\overline{\mathbf{D}} = \frac{1}{|D|} \sum_{\mathbf{d}_j \in D} \frac{\mathbf{d}_j}{\|\mathbf{d}_j\|}$$

  - Query document similarity is average over query words ccosine similarity

$$DESM(Q, D) = \frac{1}{|Q|} \sum_{q_i \in Q} \frac{\mathbf{q}_i^T \overline{\mathbf{D}}}{\|\mathbf{q}_i\| \|\overline{\mathbf{D}}\|}$$

- Extension des modèles de RI : Modèle translation de RI
  - $rsv(q,d) = \sum_{t \in q} sim(t,t').P(t'|d)$

Compares query term with every document term

$$P(t|d) = \lambda P(t|d) + \alpha \sum_{t' \in d} P(t,t'|d)P(t') +$$

$$\beta \sum_{t' \in N_t} P(t,t'|C)P(t') + (1 - \lambda - \alpha - \beta)P(t|C)$$

$$P(t,t'|d) = \frac{sim(t',t)}{\Sigma(d)} \frac{tf(t',d)}{|d|}$$

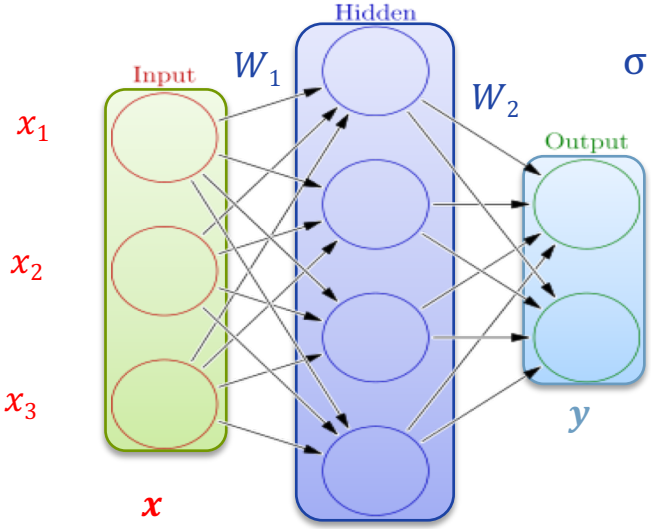
Ganguly, Roy, Mitra, and Jones. Word embedding based generalized language model for information retrieval. SIGIR 2015

Introduction à l’apprentissage profond
(Deep Learning)

Définition

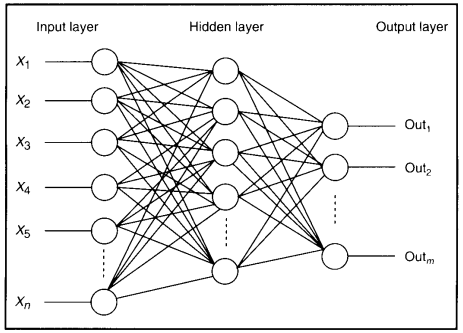
- Apprentissage profond :
  - apprentissage basé sur des réseaux de neurones
  - objectif apprendre des représentations abstraites des données
- Au cœur de cet apprentissage : → Les réseaux de neurones
  - Un RN es une succession de couches de neurones, avec une entrée (X), des couches cachées de neurones et une ou plusieurs sorties (Y) , le but est d'apprendre les « W_i » et « b_i »

$$h = \sigma(W_1 x + b_1)$$



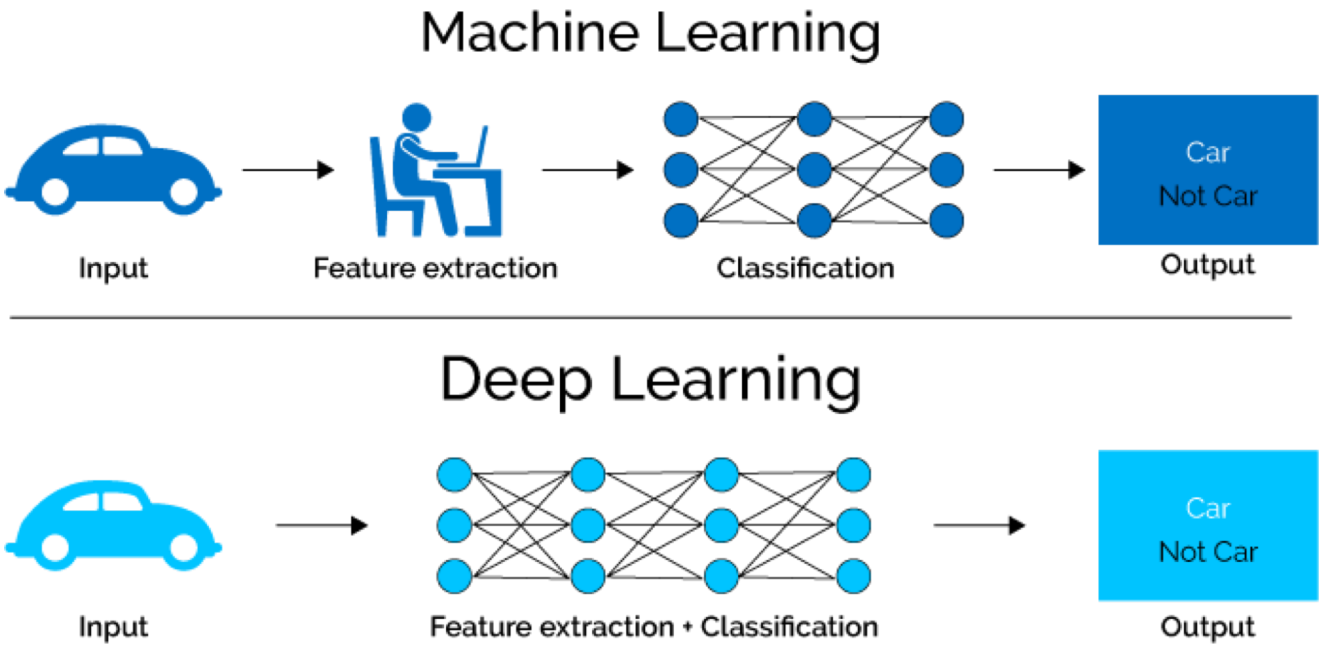
σ une fonction d'activation

$$y = \sigma(W_2 h + b_2)$$



Apprentissage de représentation

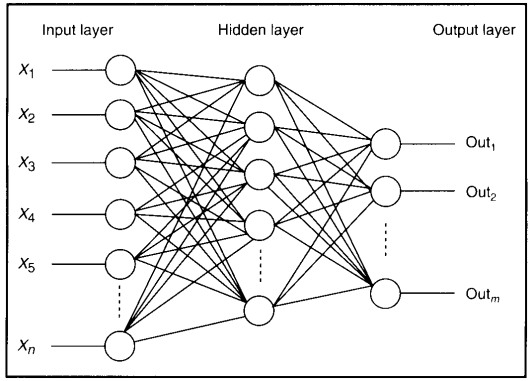
- Contrairement aux techniques d'apprentissage vues dans le cours précédent, les caractéristiques, données en entrée de l'algo d'apprentissage, sont construites automatiquement par apprentissage (on parle d'apprentissage de représentations)



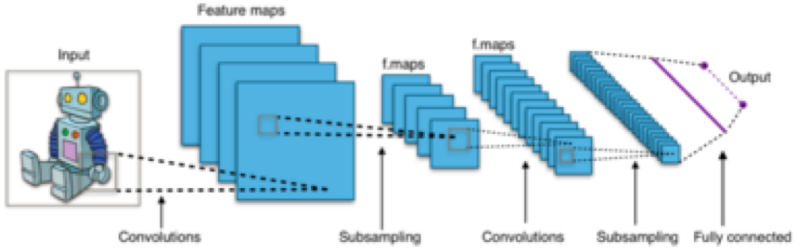
https://www.xenonstack.com/blog/static/public/uploads/media/machine-learning-vs-deep-learning.png

Modèles de réseaux de neurones

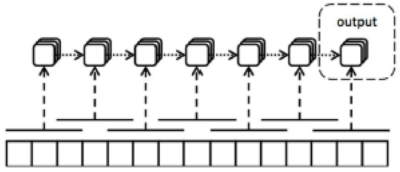
Réseaux multicouches (multilayer perceptron)

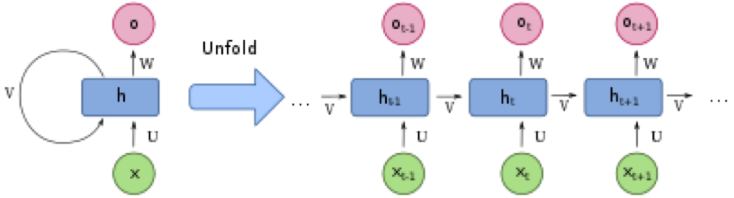


Modèles à convolution ou convolutifs (CNN)

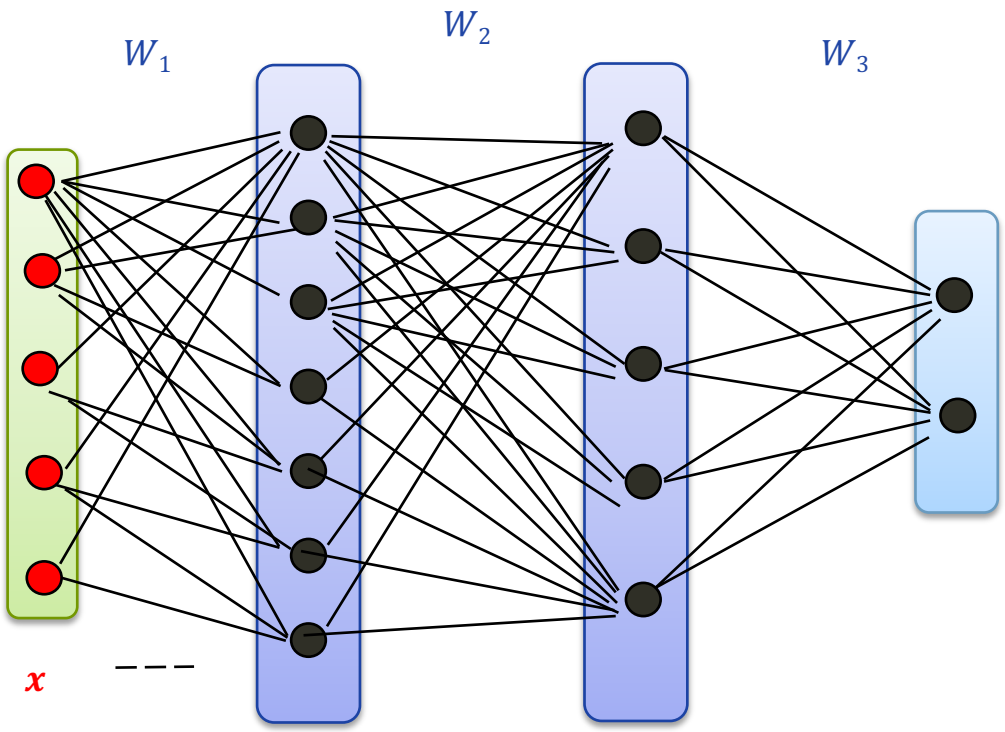


Modèles récurrents (RNN)





Réseaux multicouches (multilayer perceptron)(MLP)



σ une fonction d'activation: *tanh, relu sigmoïd, sotfmax, …*

Voir un exemple sur github

Modèles à Convolution (CNN)

These are the network parameters to be learned.

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

⋮ ⋮

Each filter detects a small pattern (3 x 3).

Modèles à Convolution (CNN)

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

Dot product →

3 ***-1***

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

If stride=2

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

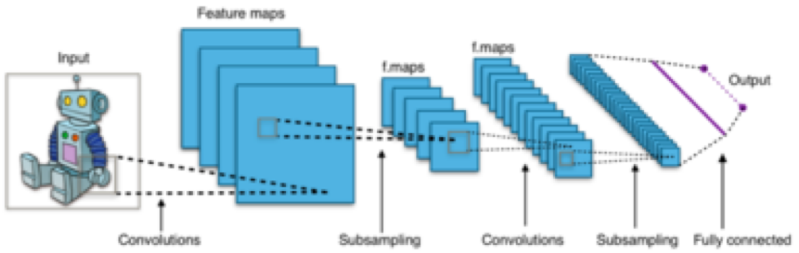
3 ***-3***

Modèles à Convolution (CNN)

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

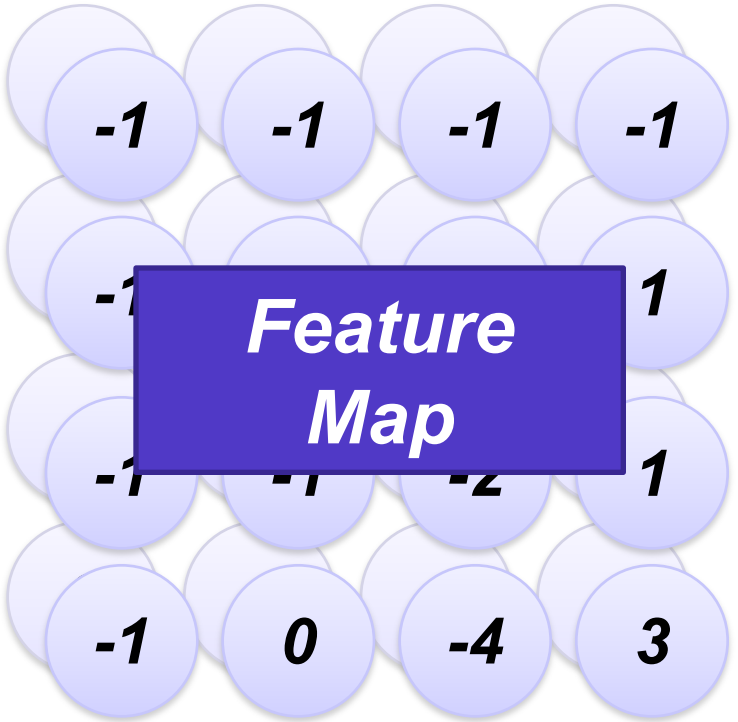
6 x 6 image



-1	1	-1
-1	1	-1
-1	1	-1

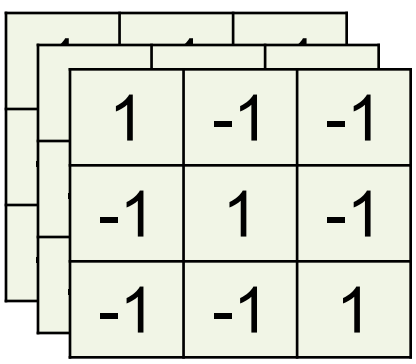
Filter 2

Repeat this for each filter

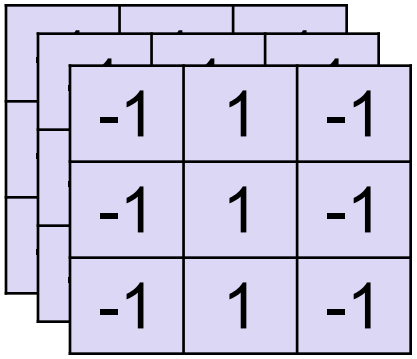


Two 4 x 4 images
Forming 2 x 4 x 4 matrix

Exemple de CNN (Color image: RGB 3 channels)

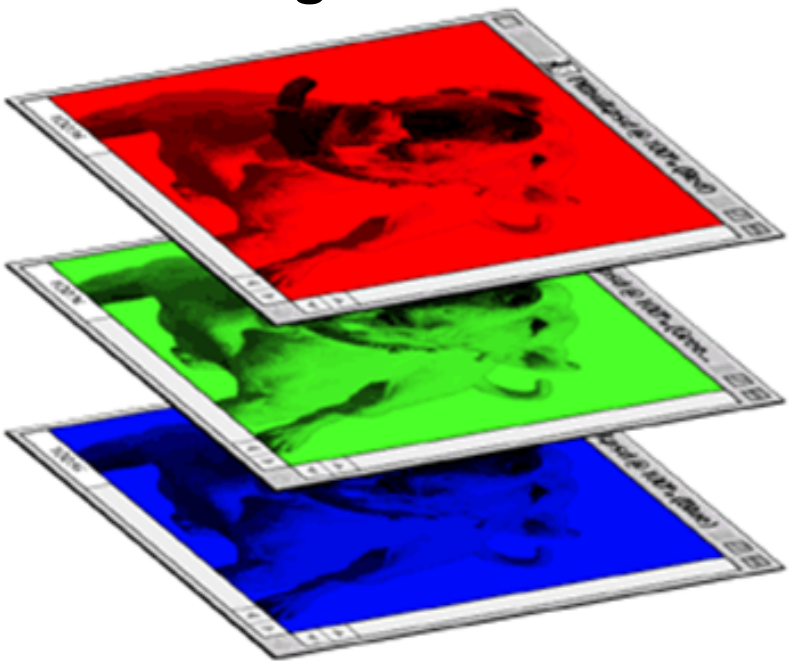


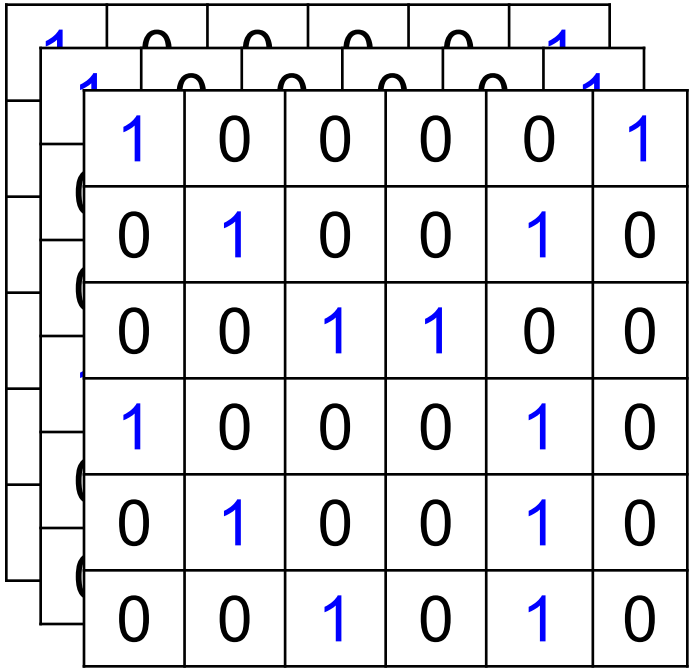
Filter 1



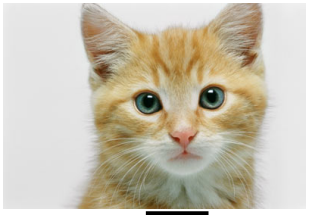
Filter 2

Color image





Un Exemple de CNN complet



Can repeat many times

Convolution

Max Pooling

Convolution

Max Pooling

Flattened

cat dog .....

Fully Connected Feedforward network

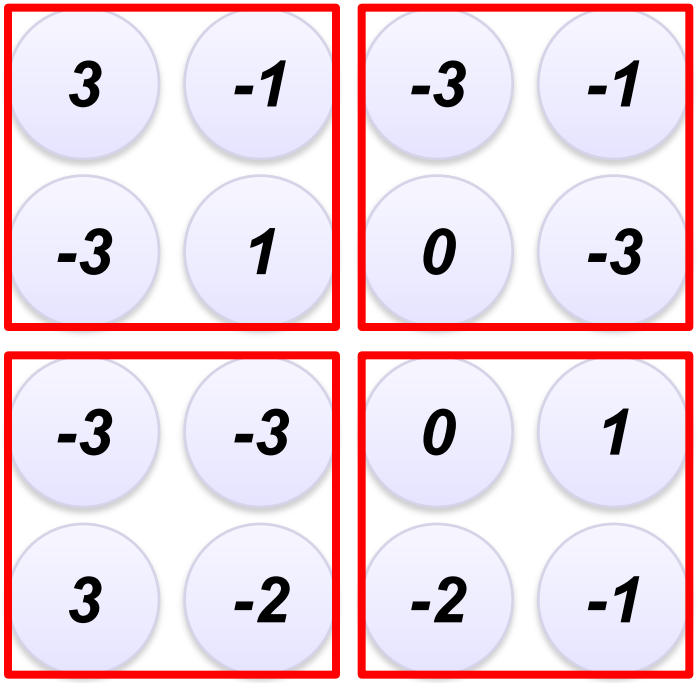
Max Pooling

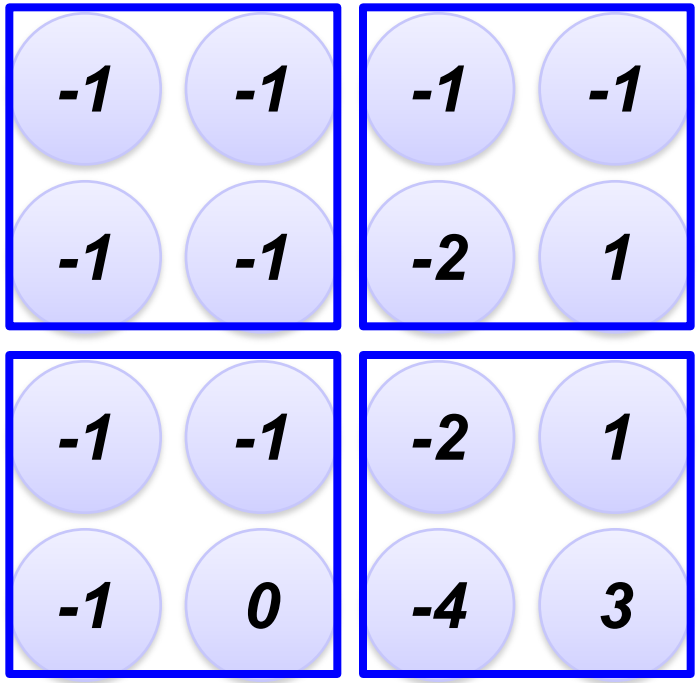
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

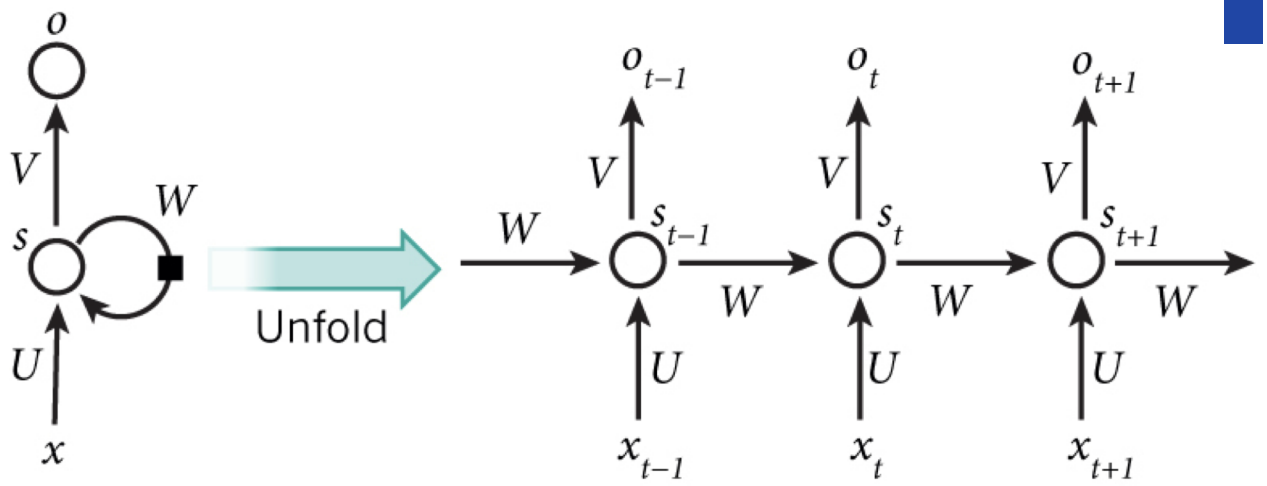
-1	1	-1
-1	1	-1
-1	1	-1

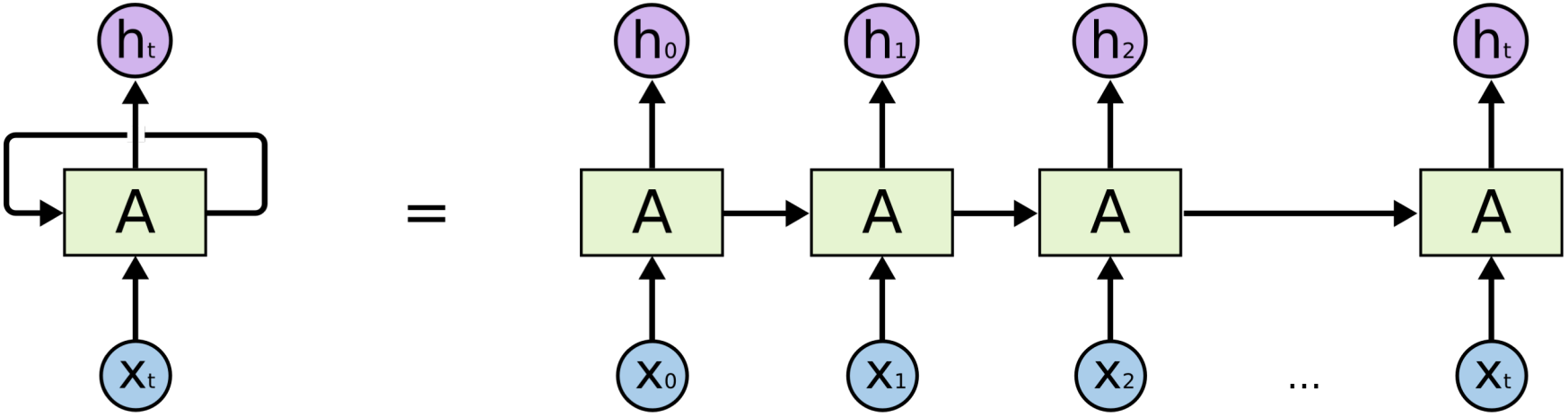
Filter 2





Modèle de RN récurrents (Recurrent Neural Network) (RNN)

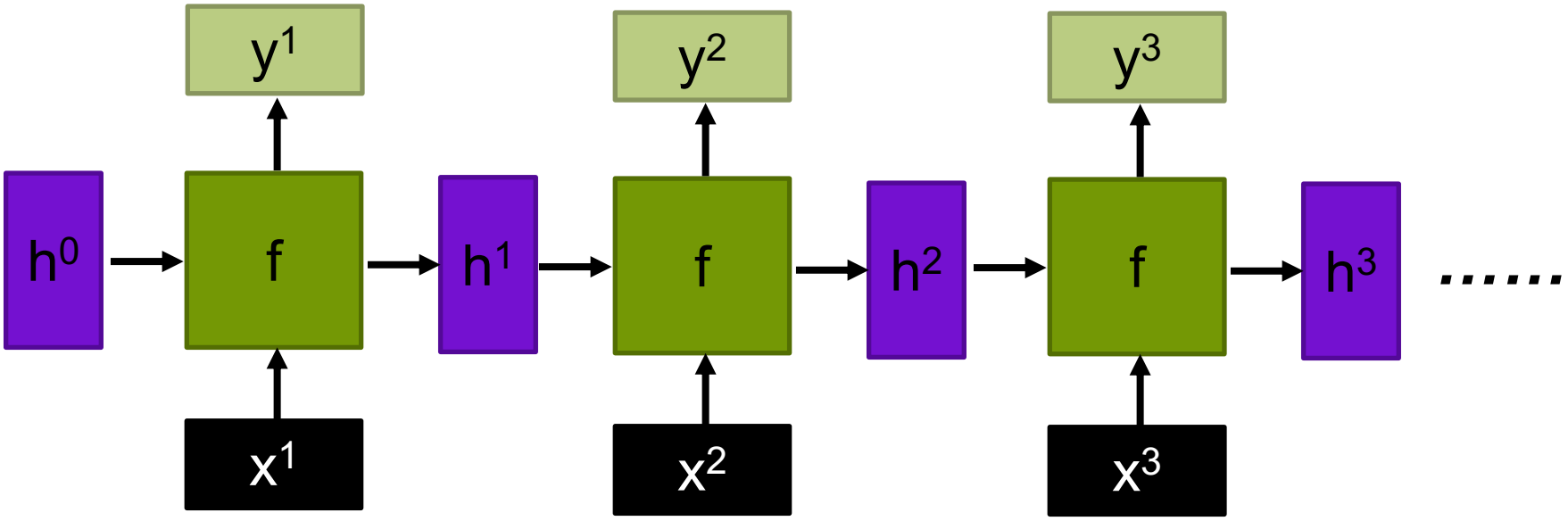




Modèle de RN récurrents (RNN)

- Given function f: $h^i, y=f(h,x)$

h^i are vectors with the same dimension



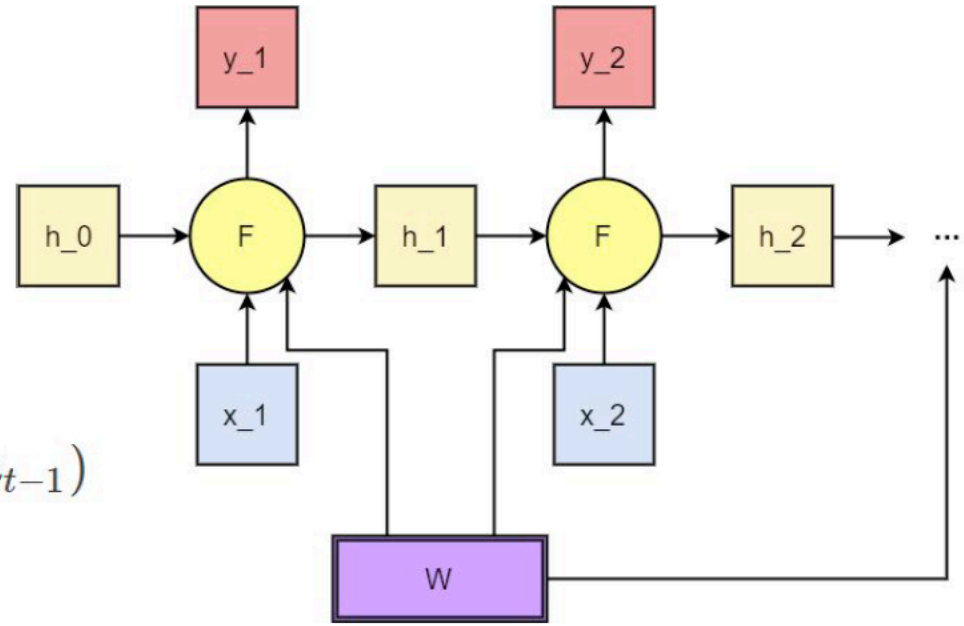
No matter how long the input/output sequence is, we only need one function f.

Un RNN simple (Naif)

RNN Diagram

… with shared (tied) weights

$$(h_1, y_1) = F(h_0, x_1, W)$$
$$(h_2, y_2) = F(h_1, x_2, W)$$

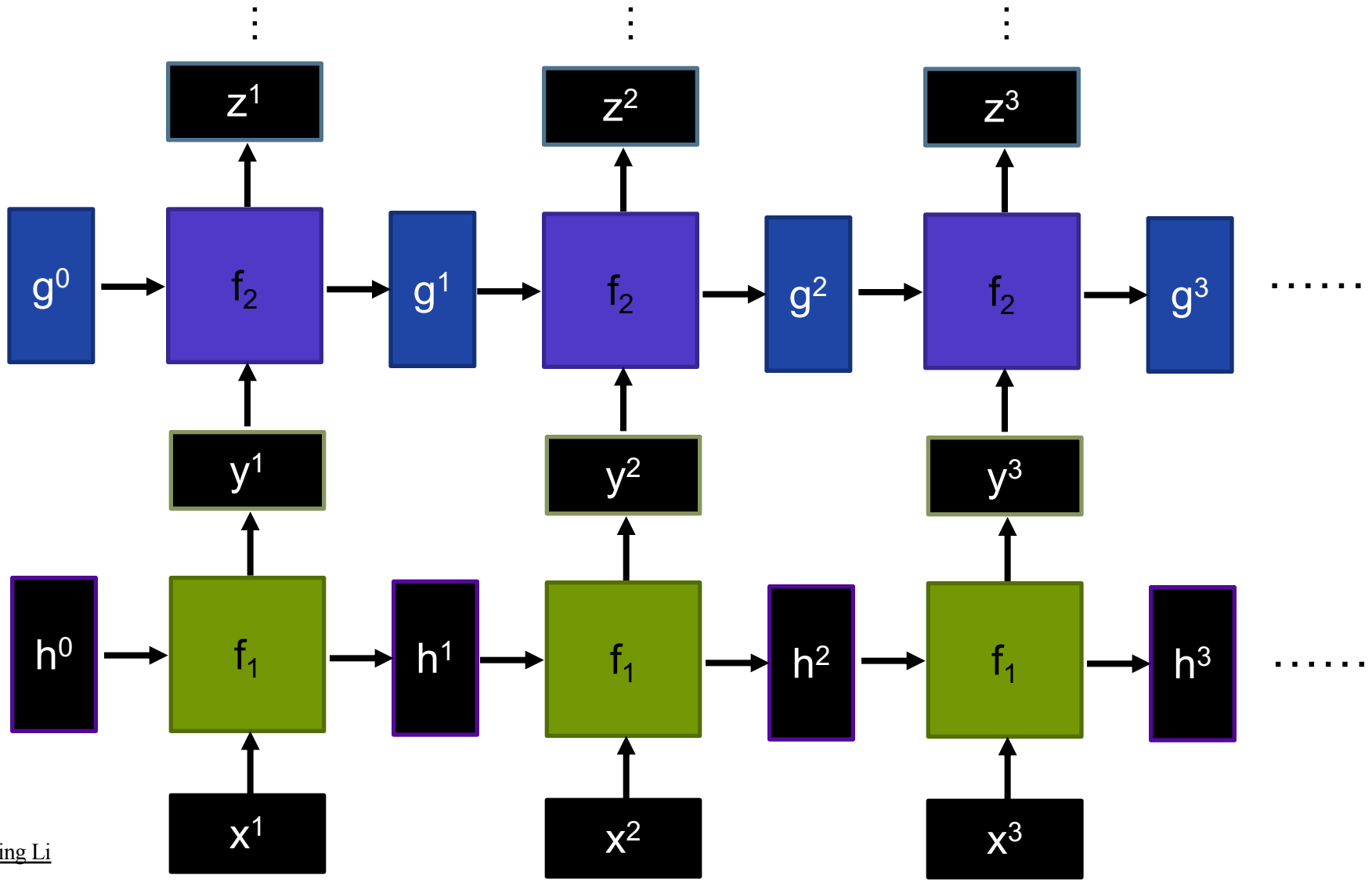


A simple implementation of F

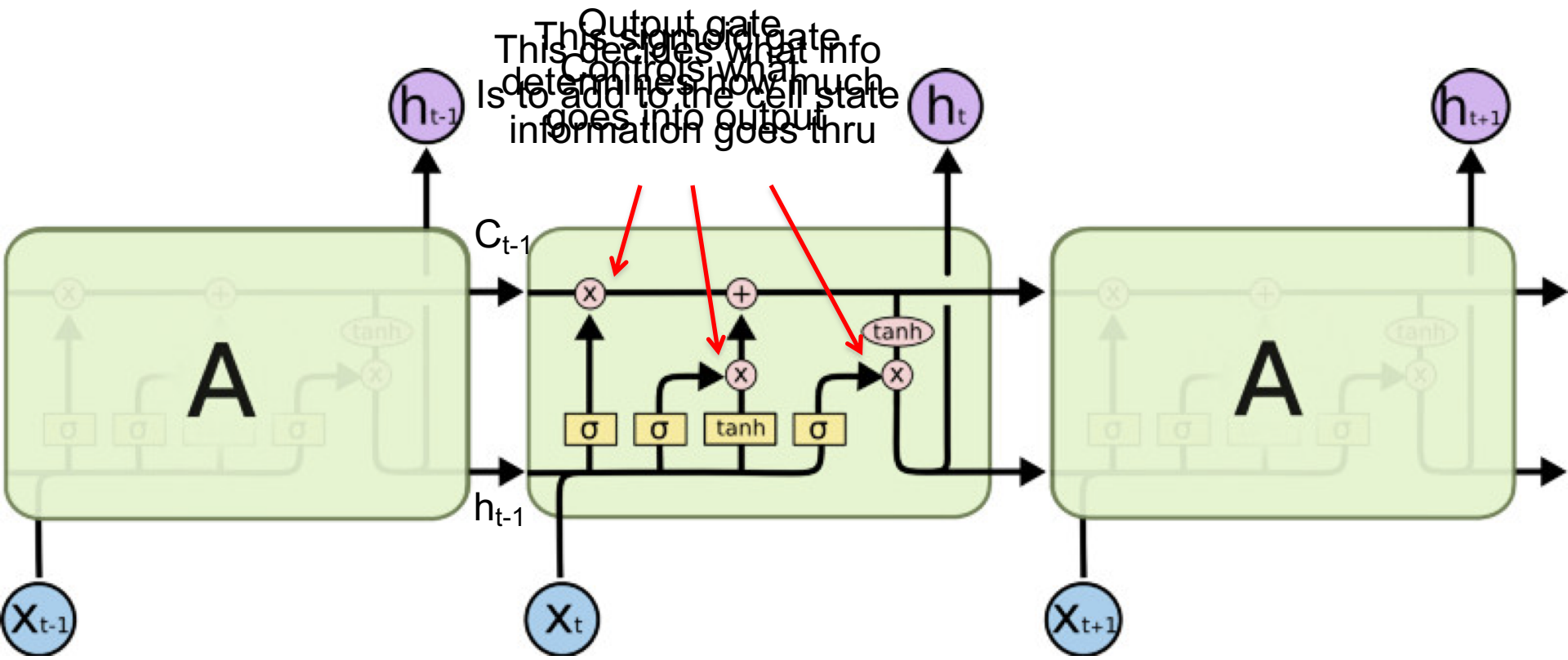
$$h_t = tanh(W_{ih}x_t + W_{hh}h_{t-1})$$
$$y_t = W_{ho}h_t$$

Un RNN Profond

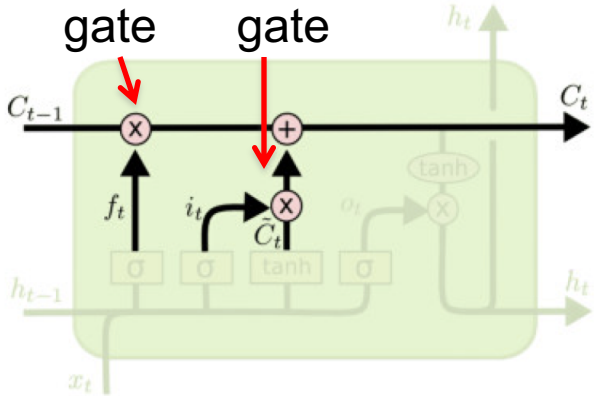
$h^i, y = f_1(h, x),\ g^i, z = f_2(g, y)$



Un RNN type LSTM
(Long Short Term memory)

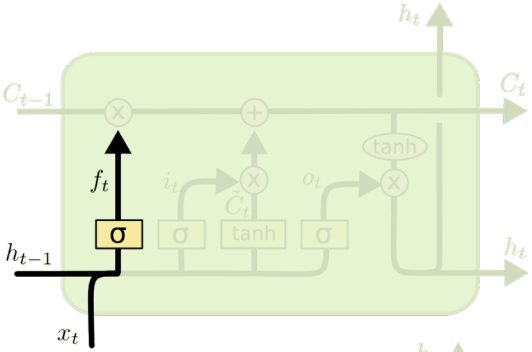


Forget gate input gate

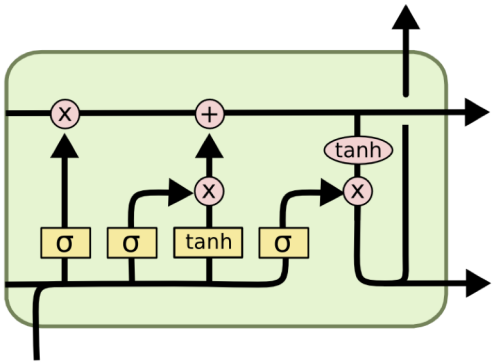


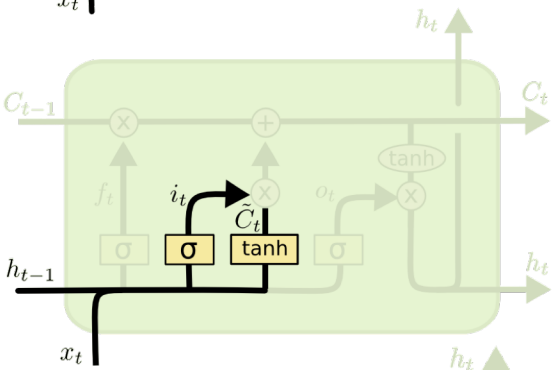
The core idea is this cell state C_t, it is changed slowly, with only minor linear interactions. It is very easy for information to flow along it unchanged.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



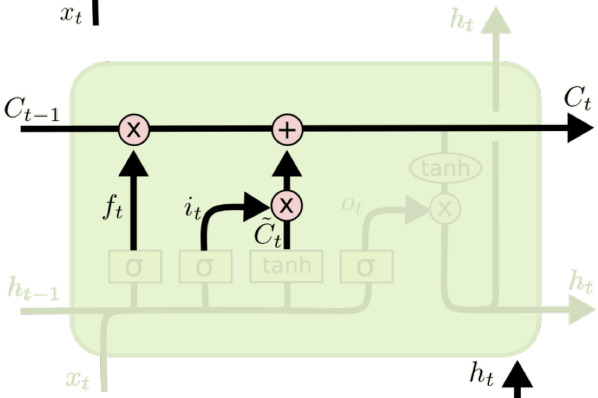
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$





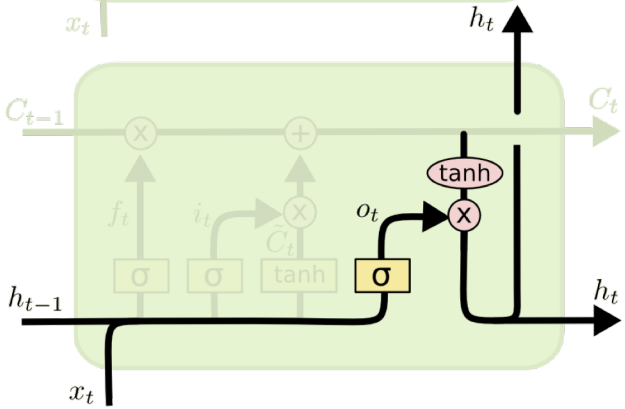
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

i_t decides what component is to be updated. C'$_t$ provides change contents



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

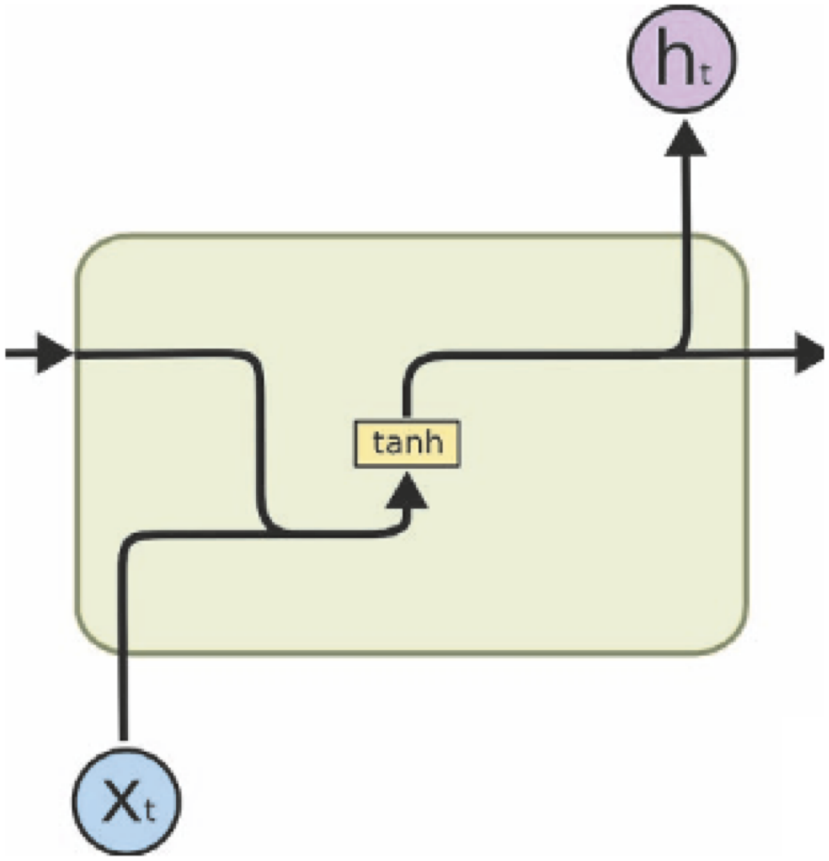
Updating the cell state



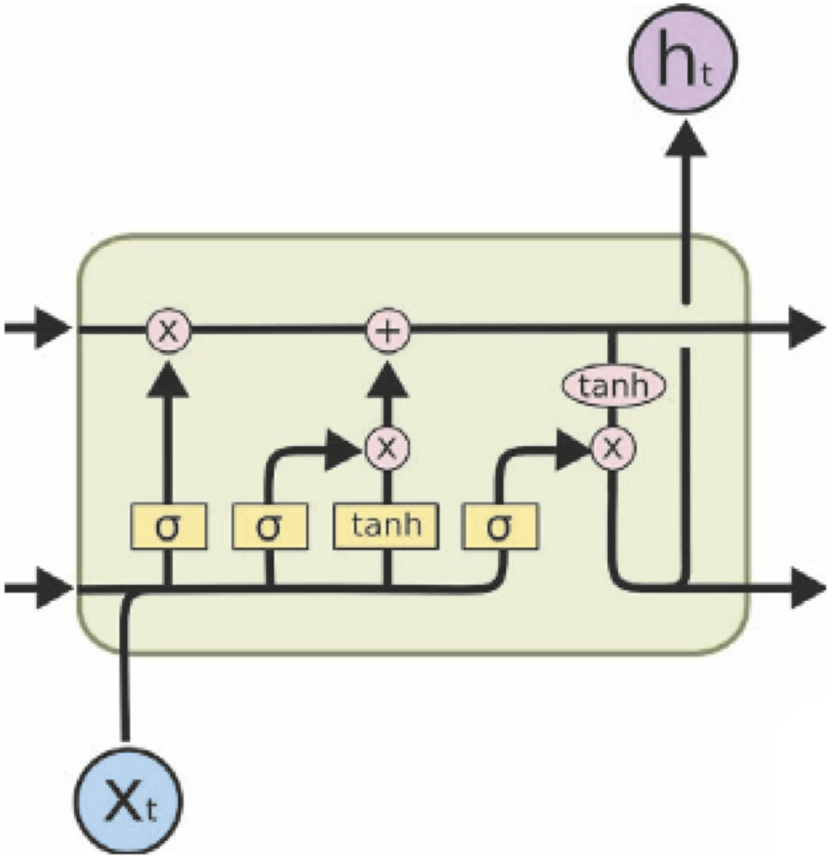
$$o_t = \sigma\left(W_o \; [\, h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Decide what part of the cell state to output

RNN naif vs LSTM

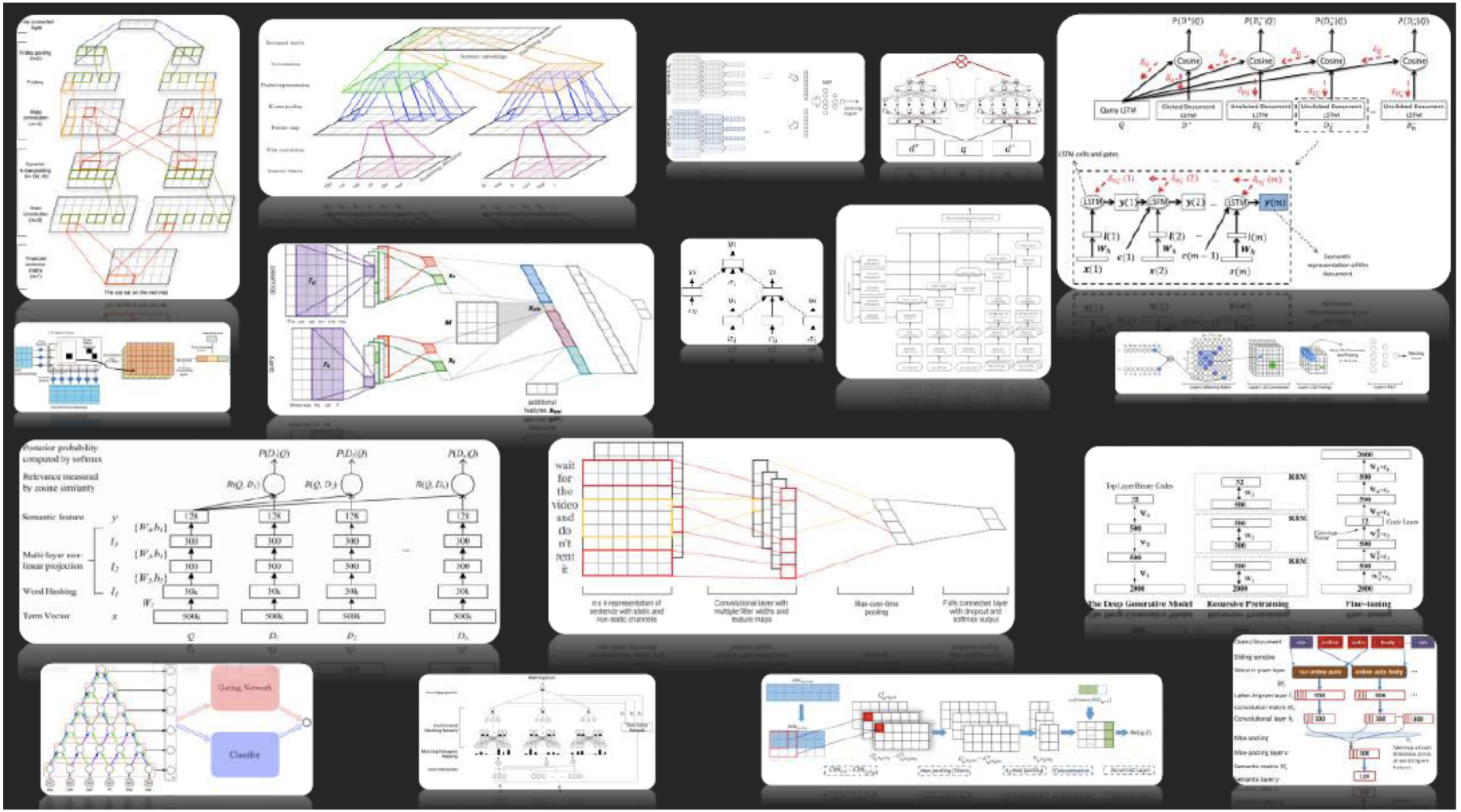


(a) RNN



(b) LSTM

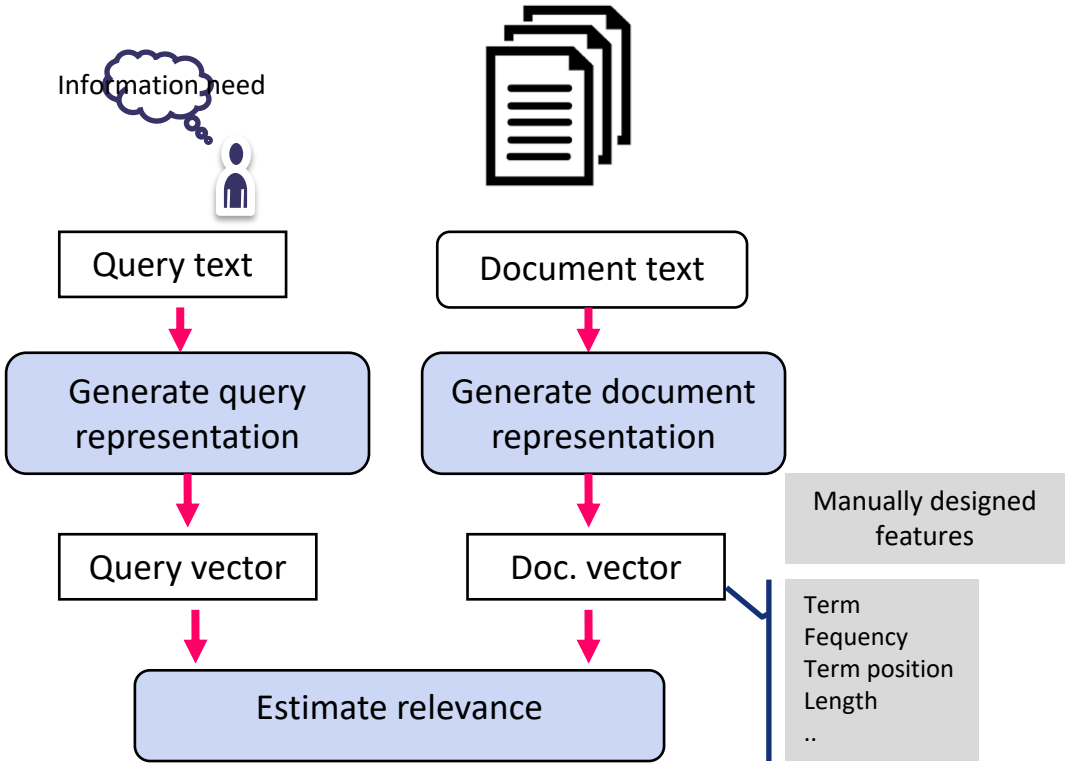
Multitude de modèles en RI



Modèles neuronaux de RI
Deep Neural IR

Fonctions principales d'un système de RI

- Information rerieval (IR) in three simple steps:
  - Generate query representation
  - Generate doc. representation
  - Estimate relevance

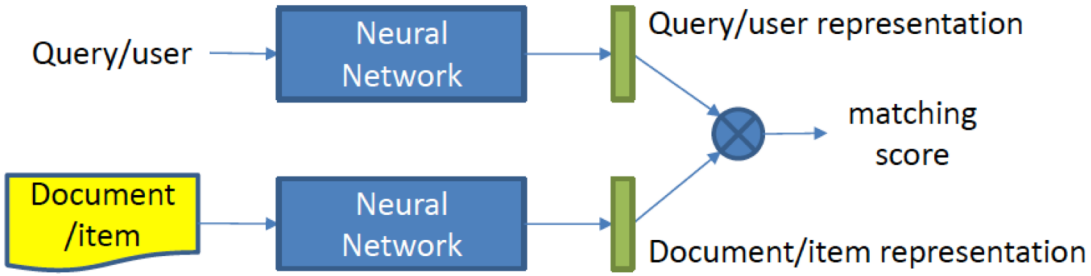


$$\sum_{w \in q \cap d} \ln \frac{N - df(w) + 0.5}{df(w) + 0.5} \cdot \frac{(k_1 + 1) \times c(w,d)}{k_1((1-b) + b\frac{|d|}{avdl}) + c(w,d)} \cdot \frac{(k_3 + 1) \times c(w,q)}{k_3 + c(w,q)}$$

Learning 2 Rank (SVM, NN, ..)

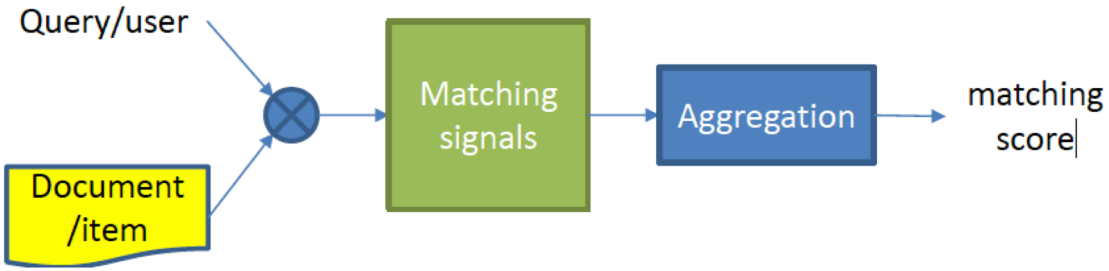
Les deux classes de modèles neuronaux de RI

- Apprentissage ds représentations Learning (Representation-based models)



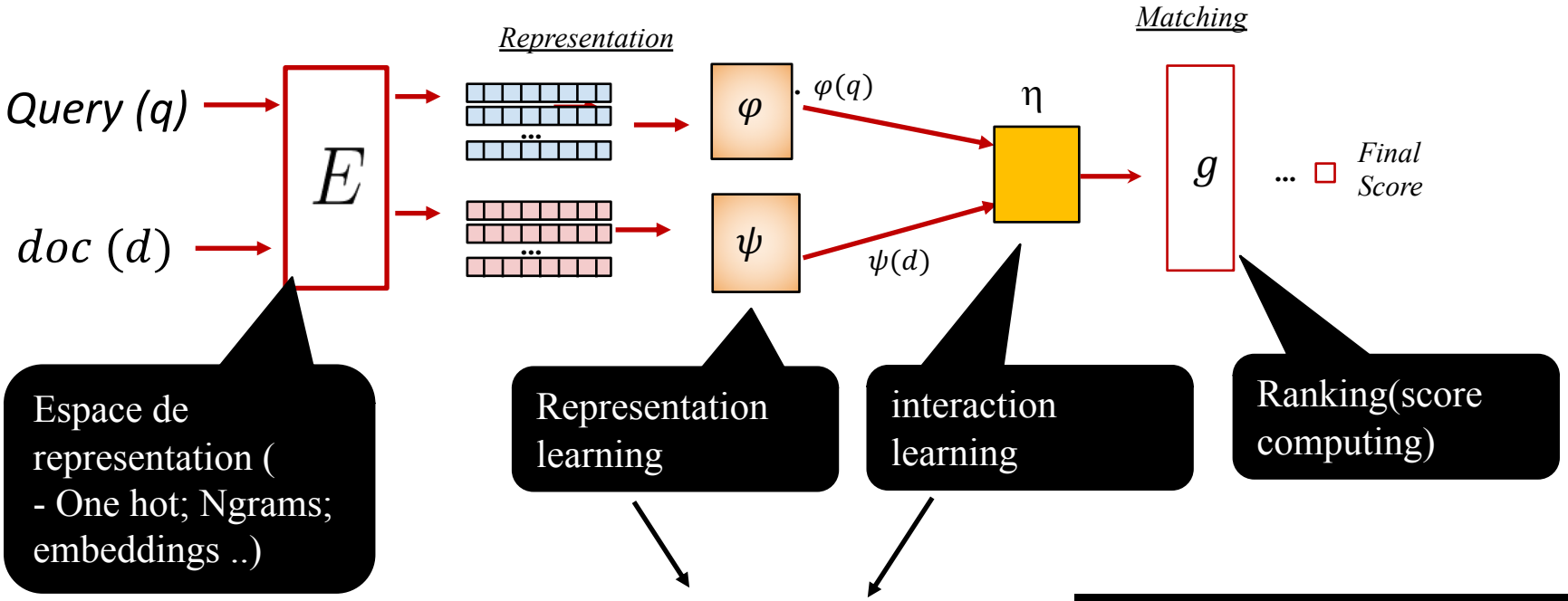
- Learn latent patterns that better represent a document and a query
- matching function is simple (cosine, MLP)

- Apprentissage de la fonction de «matching » (Interaction-based models)



Learn relevance from interactions between a document and a query rather than from individual representations

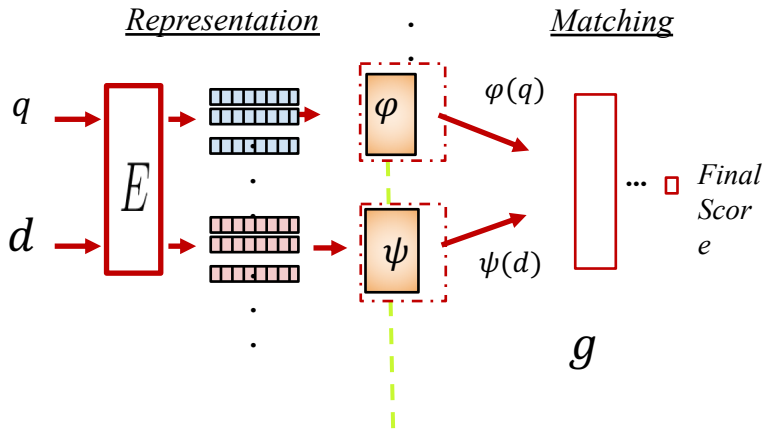
Architecture générale



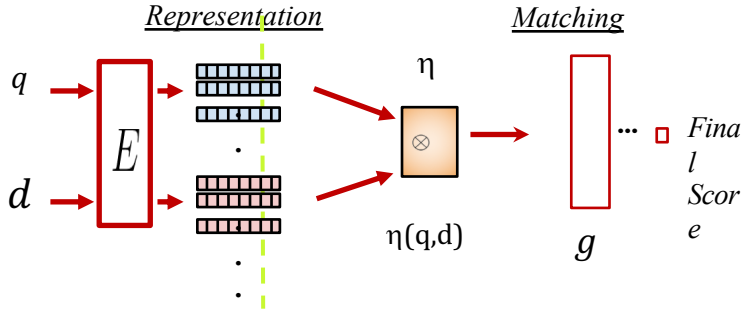
Modèles de la littérature

Neural text matching (Guo et al., 2016)

Representation Focused
- DSSM (Huang et al., 2013)
- ARC-I (Hu et al. 2014)
- CLTR (Severyn et al., 2015)
- LSTM-RNN (Palangi rt al. 2016)
- DRCN (Kim et al., 2019)

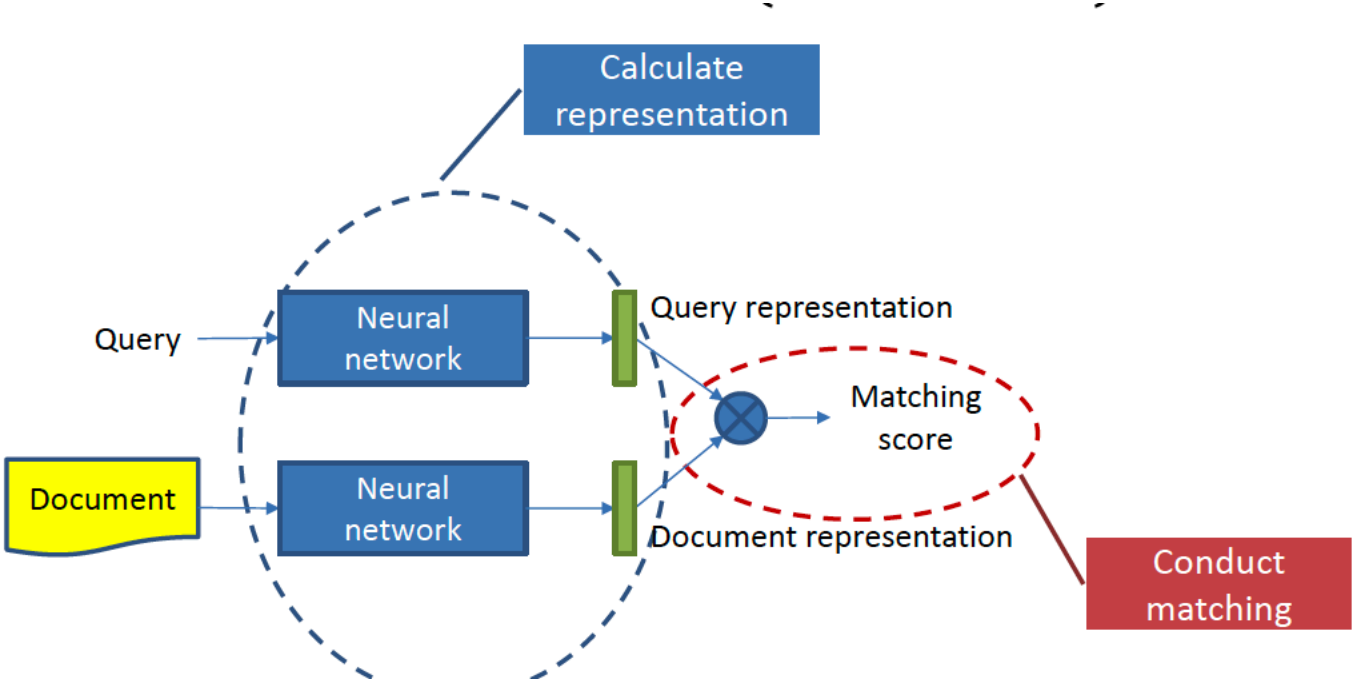


Interaction Focused
- ARC-II (Hu et al., 2014)
- MatchPuramid (Pang et al. 2016)
- DRMM (Guo et al., 2016)
- DUET (Mitra et al., 2017)

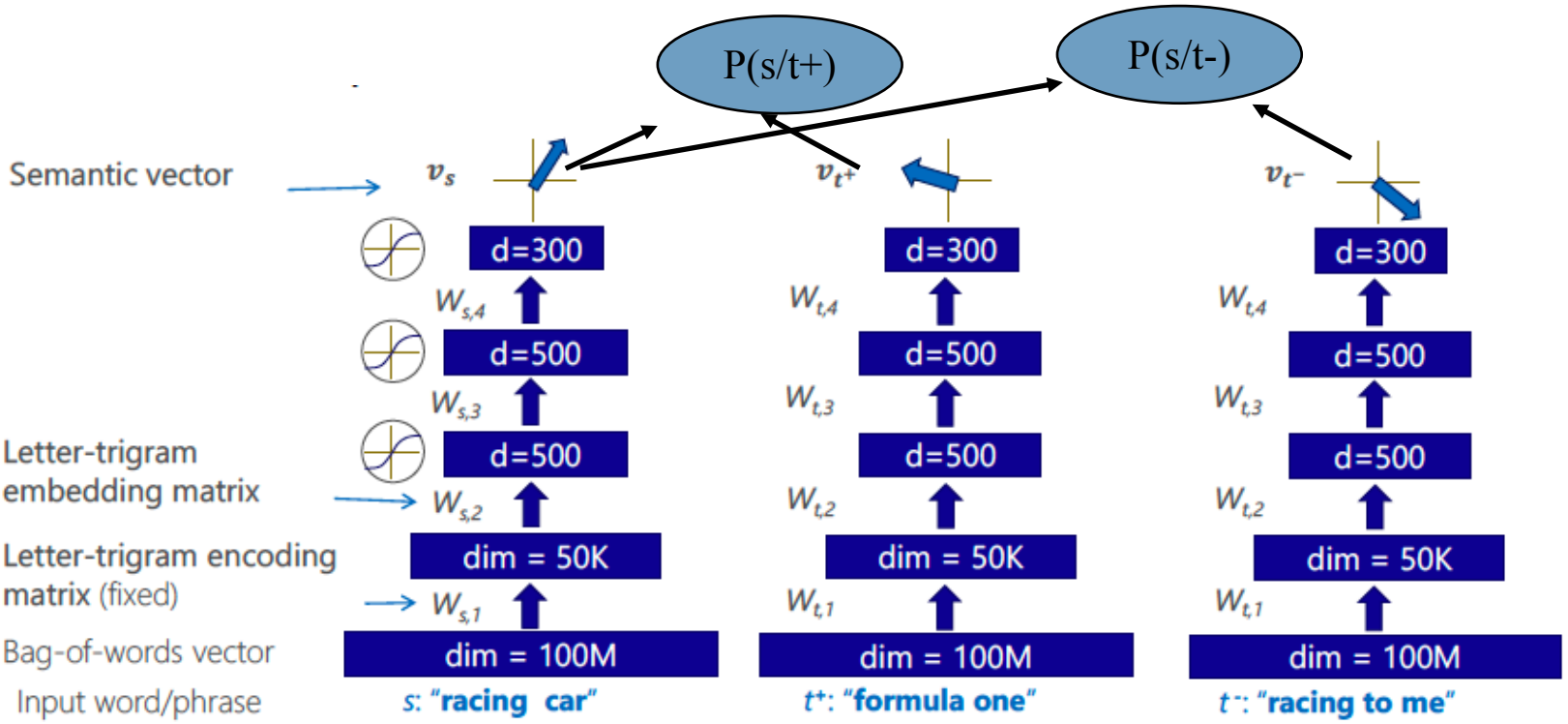


Apprentissage de représentations

- Étape 1 : Calculer (apprendre) les représentations des textes (requête et document) $\phi(x)$
- Étape 2 : Calculer leur similarité (cosine, …) $F(\phi(q), \phi(d))$



Le modèle DSMM (autoencoder)



DSSM [Huang et al., 2013, Shen et al., 2014)

© He et al., CIKM '14 tutorial

Learn a fixed-dimensional vector representation for each text separately and then perform matching within the latent space

Le modèle DSMM (autoencoder)

- DSSM Model : (Autoencoder)

DSSM Matching Function

- Cosine similarity between semantic vectors

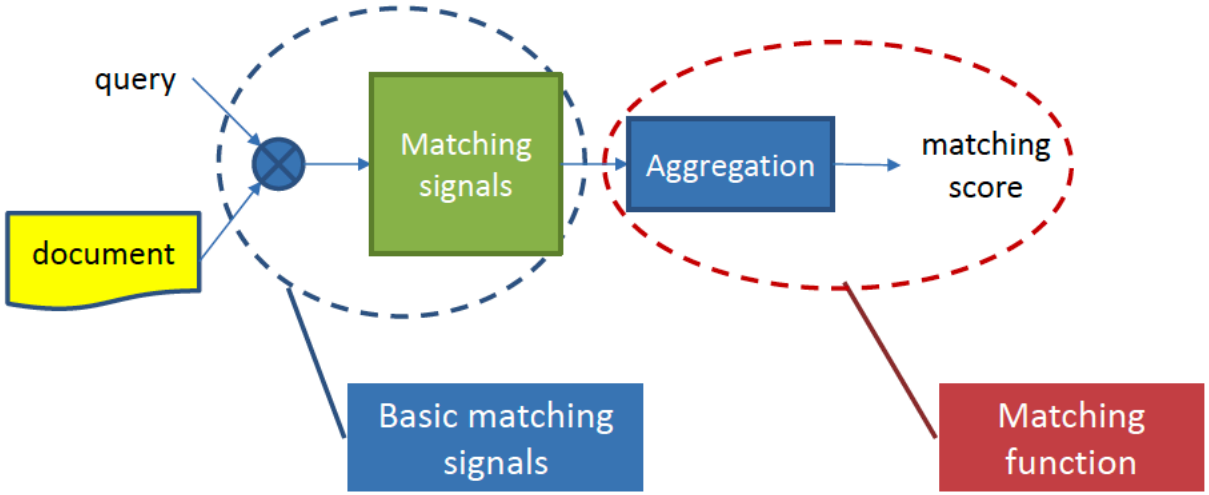
$$S = \frac{x^T \cdot y}{|x| \cdot |y|}$$

- Training
  - A query q and a list of docs $D = \{d^+, d_1^-, \cdots, d_k^-\}$
  - d^+ positive doc, $d_1^-, \cdots, d_k^-$ negative docs to query
  - Objective:

$$P(d^+|q) = \frac{\exp(\gamma \cos(q, d^+))}{\sum_{d \in D} \exp(\gamma \cos(q, d))}$$

Apprentissage de la fonction de matching

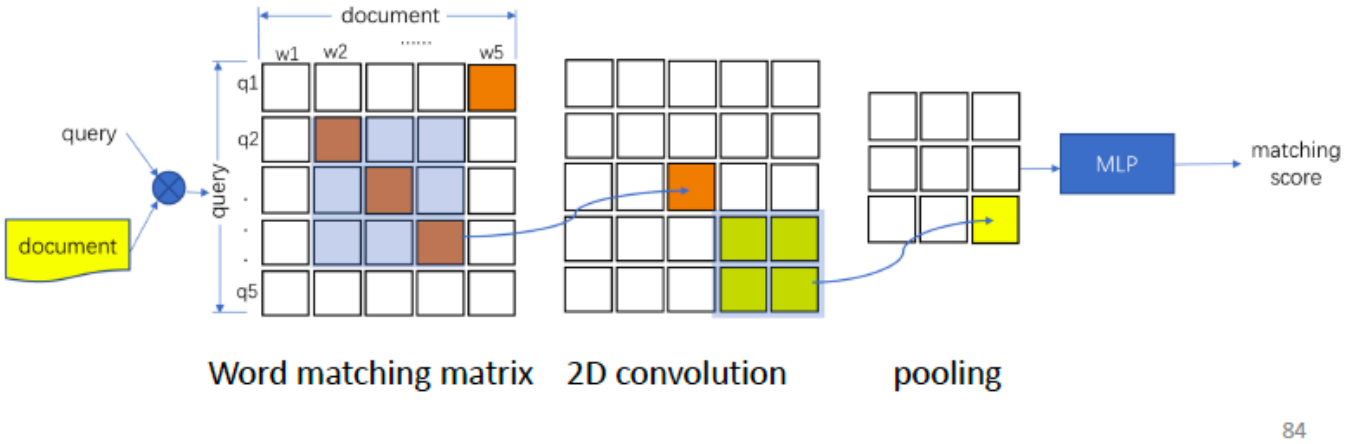
- Étape 1 : construire des signaux de base de bas niveau
- Étape 2 : agréger les modèles de correspondance (matching)



Interaction-based models compute the interaction between each individual term of both query and documents.

Modèle MatchPyramid

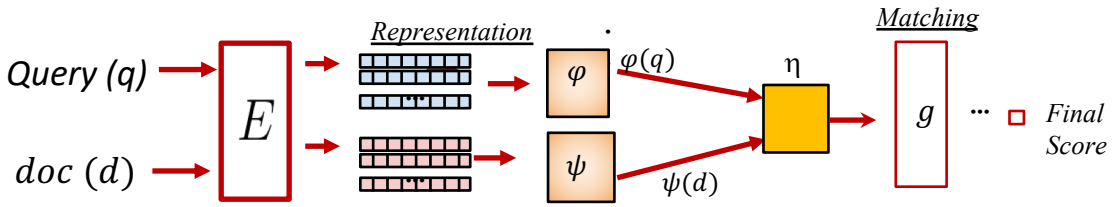
- Inspired by Image recognition
- Basic matching signals: word level matching matric
- Matching function: 2D convolution+ MLP

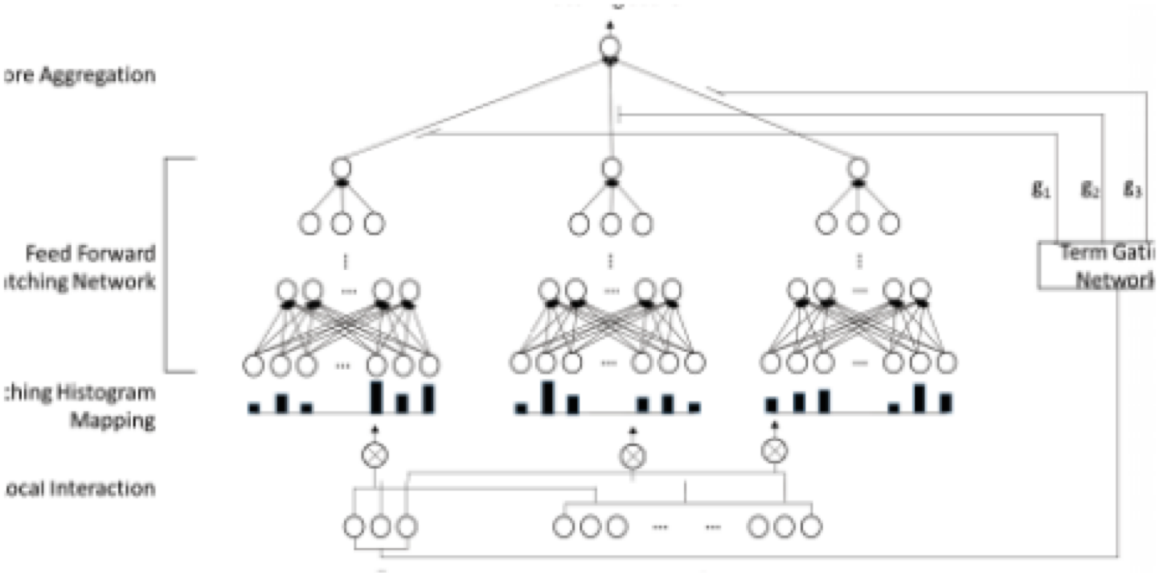


(Peng et al. AAA'16)

Modèle hybride (Hybrid model)

- Representation learning+ matching function learning

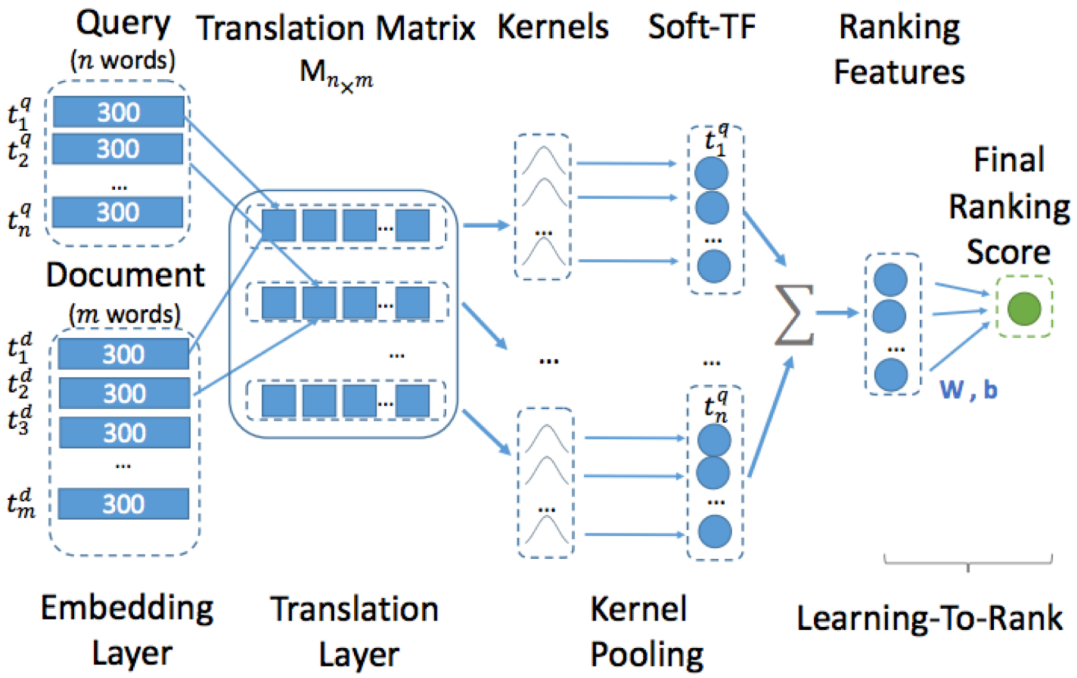




(Guo et al., 2016)

KNRM

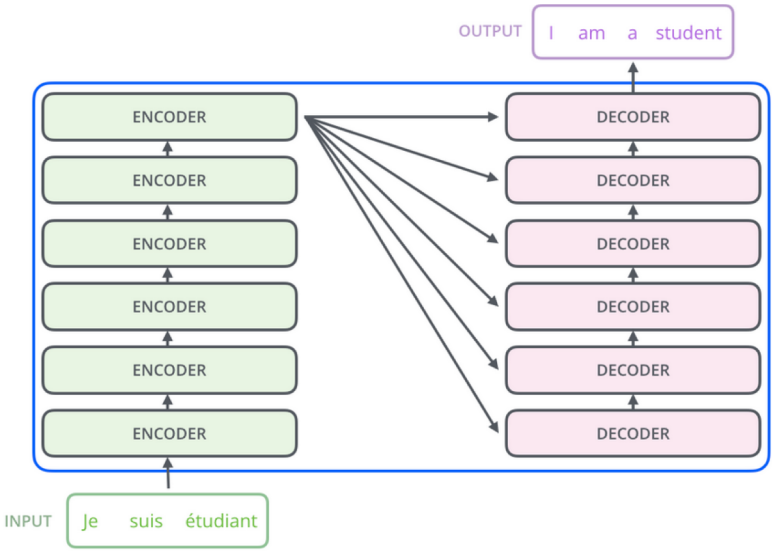
- Compute word-interaction matrix, (s-cosine similarity of word embeddings)
- This results in k-dimensional vector.
- Aggregate the query term vectors into a fixed-dimensional query representation.

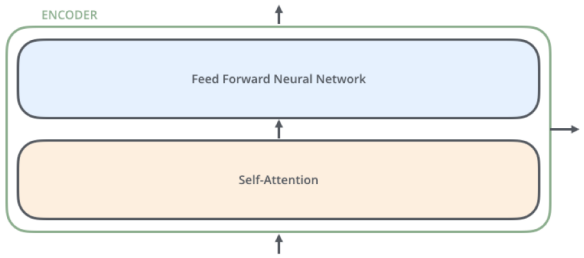


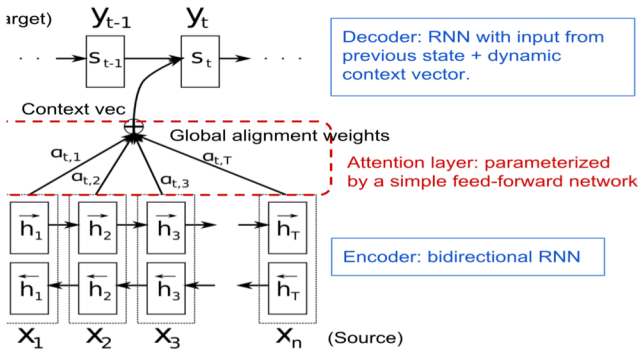
K-NRM [Xiong et al., 2017b]

Deep learning and IR
La révolution des transformeurs à la BERT

- Transformeurs (*transformers*) ou (modèle auto attentif)(Self attention)
  - Catégorie de Deep ML utile pour le traitement de la langue (texte, séquences de mots) qui ne nécessite pas le traitement de la sequence dans l'ordre des mots (contrairement au RNN).
  - BERT, ELMO, ULMfit, OpenAI GPT
- Sont construits autour de trois notions:
  - Encodeur, Décodeur, Attention(self)







Exemples de transformeurs

- Un panoplie de *transformeurs* avec des architectures différentes : (Codeur-self attention avec ou sans(décodeur))

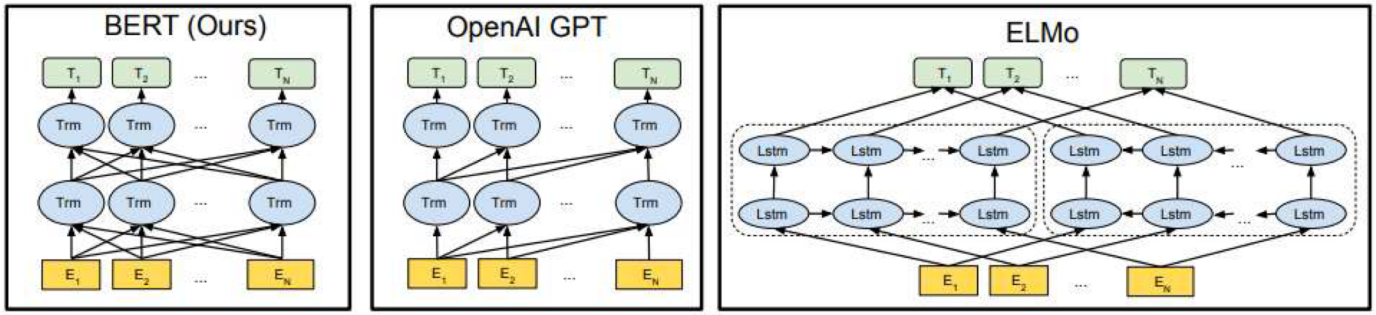
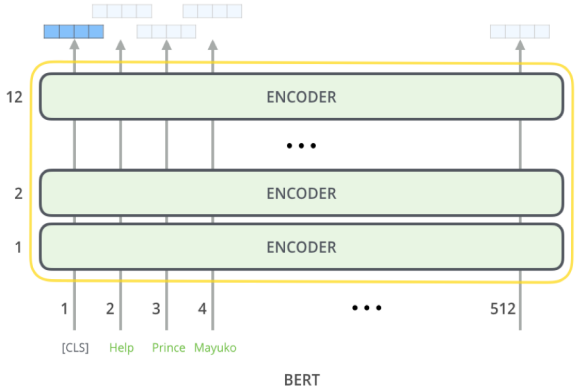
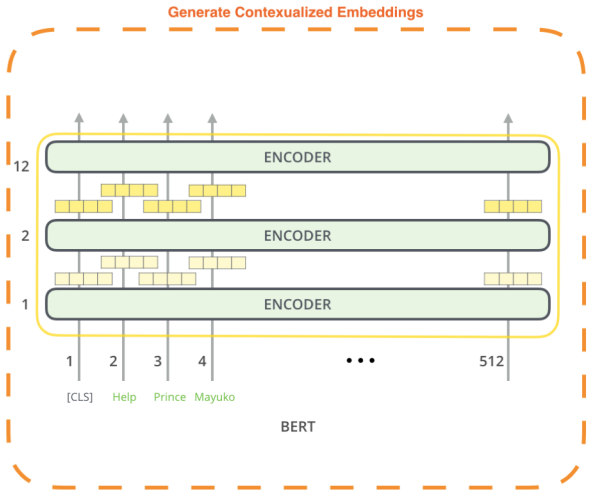
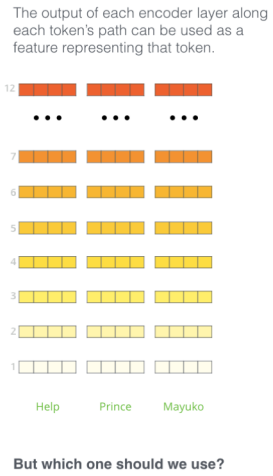


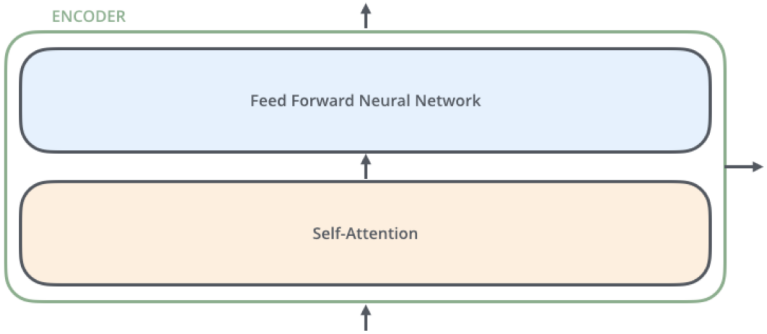
Figure 1: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTM to generate features for downstream tasks. Among three, only BERT representations are jointly conditioned on both left and right context in all layers.

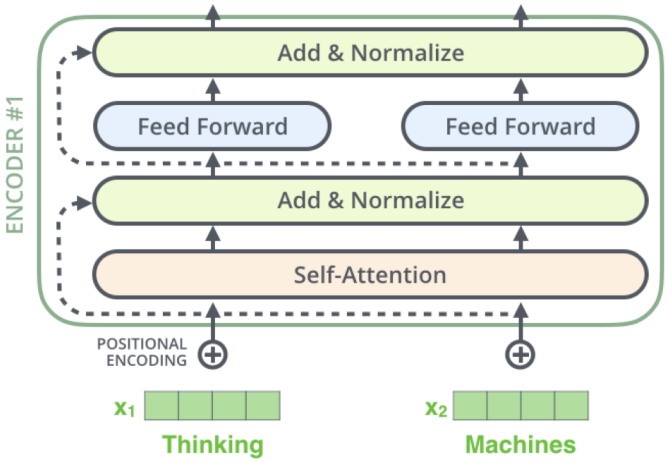
Encodeur (*Encoder*)









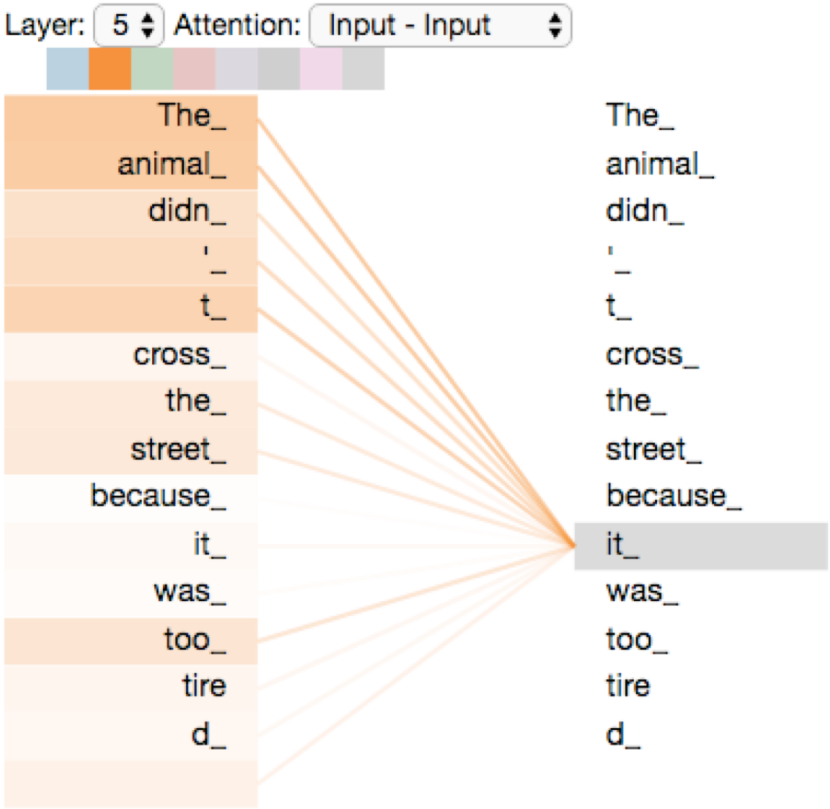


© *alammar.github.io/*

Auto attention (*Self attention*)

"The animal didn't cross the street because it was too wide"

" The animal didn't cross the street because it was too tired"

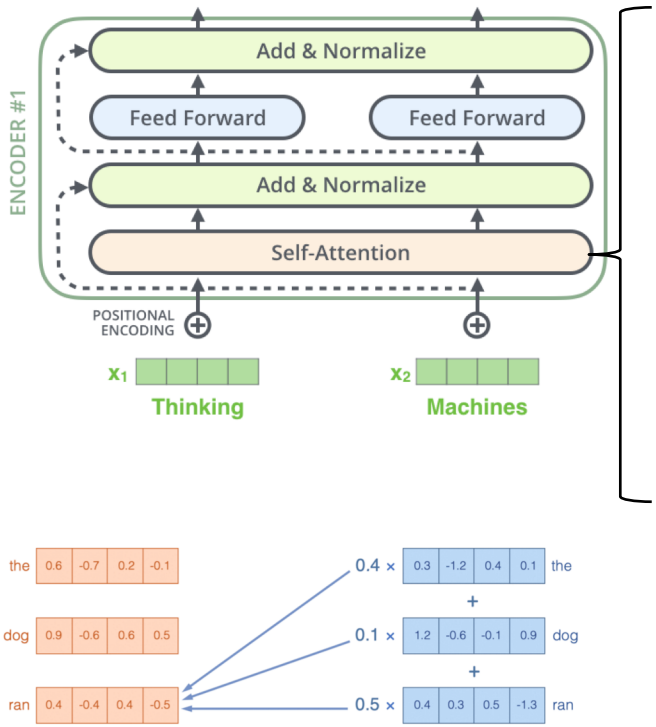


As the model processes each word (each position in the input sequence), self-attention allows it to look at other positions in the input sequence for clues that can help lead to a better encoding for this word

© blog de http://jalammar.github.io/

Transformeurs (*Tranformers*)

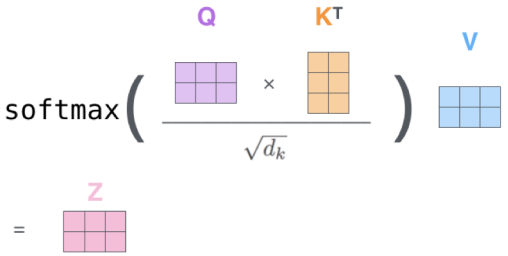
Auto attention (*Self Attention*)



Query

Key

Value



© alammar.github.io/

Self-attention

https://arxiv.org/abs/1706.03762

q: query (to match others)

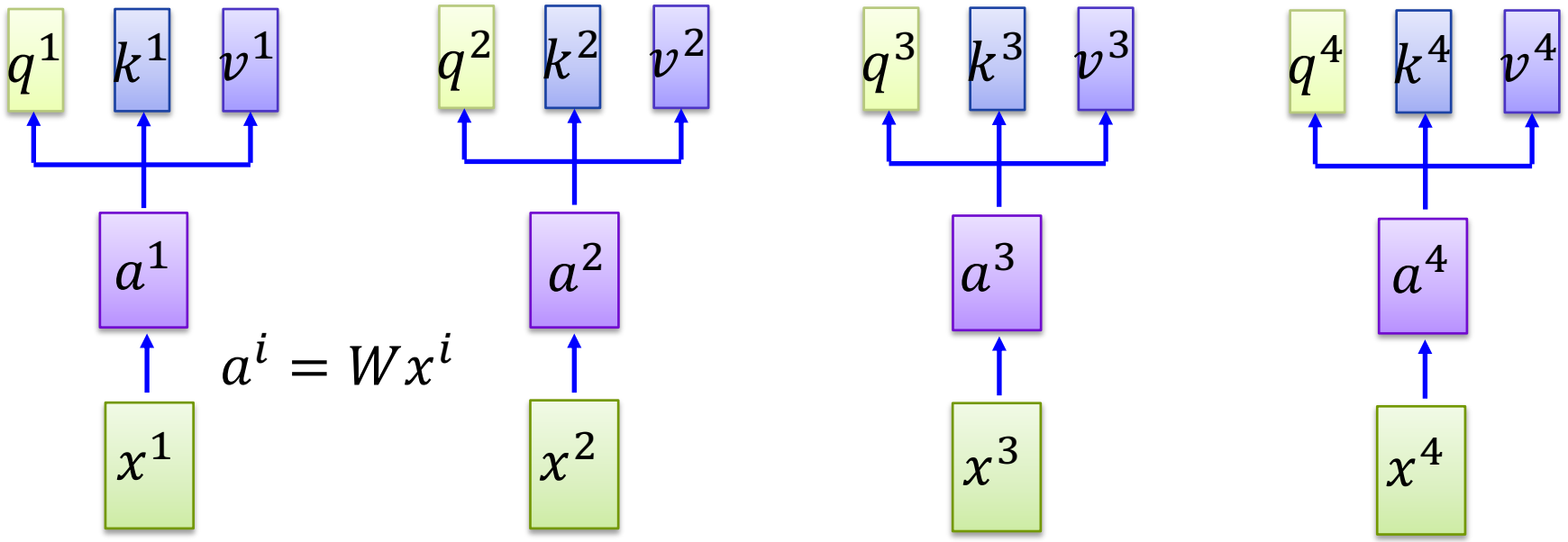
$$q^i = W^q a^i$$

k: key (to be matched)

$$k^i = W^k a^i$$

v: information to be extracted

$$v^i = W^v a^i$$

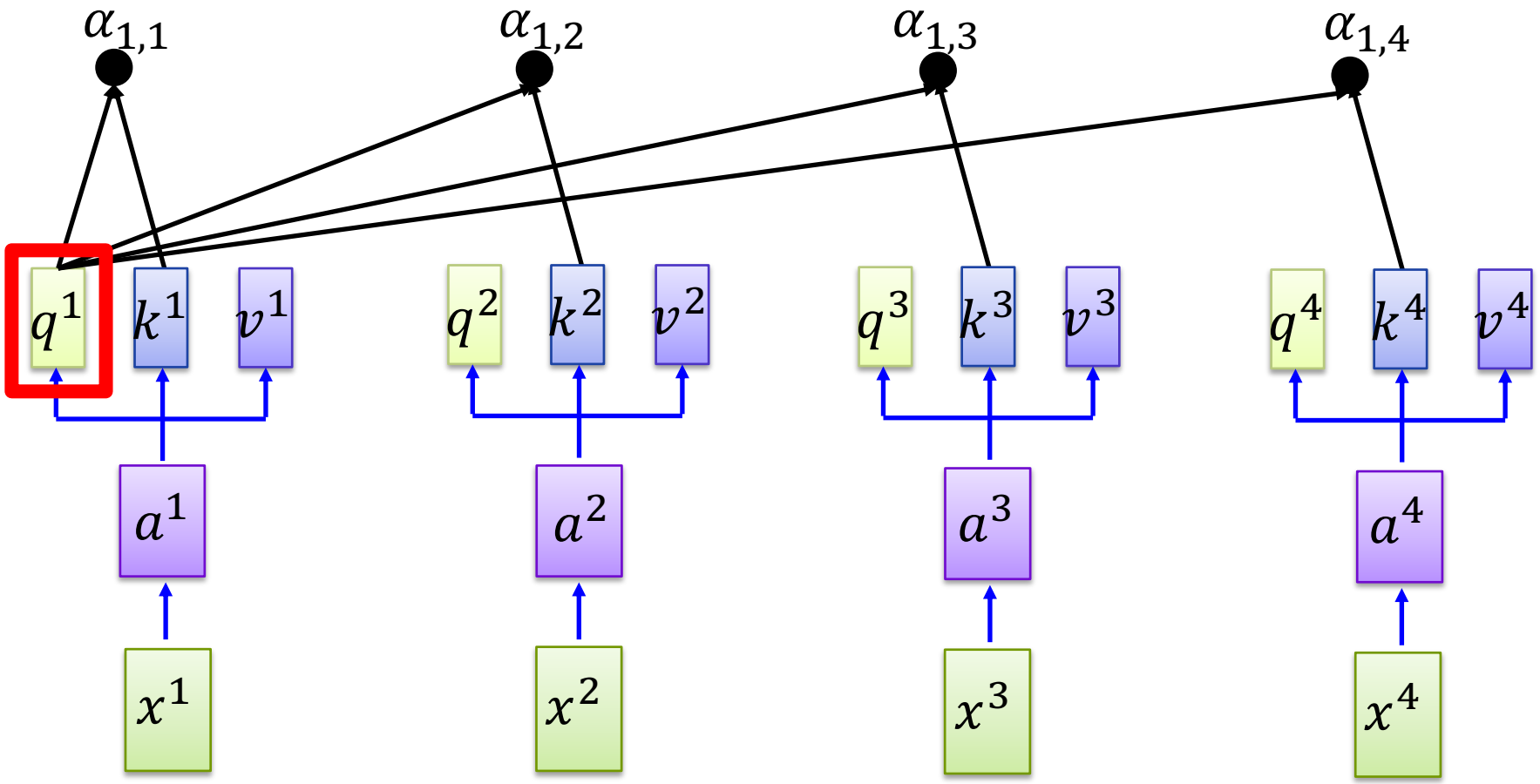


Self-attention

query q key k attention

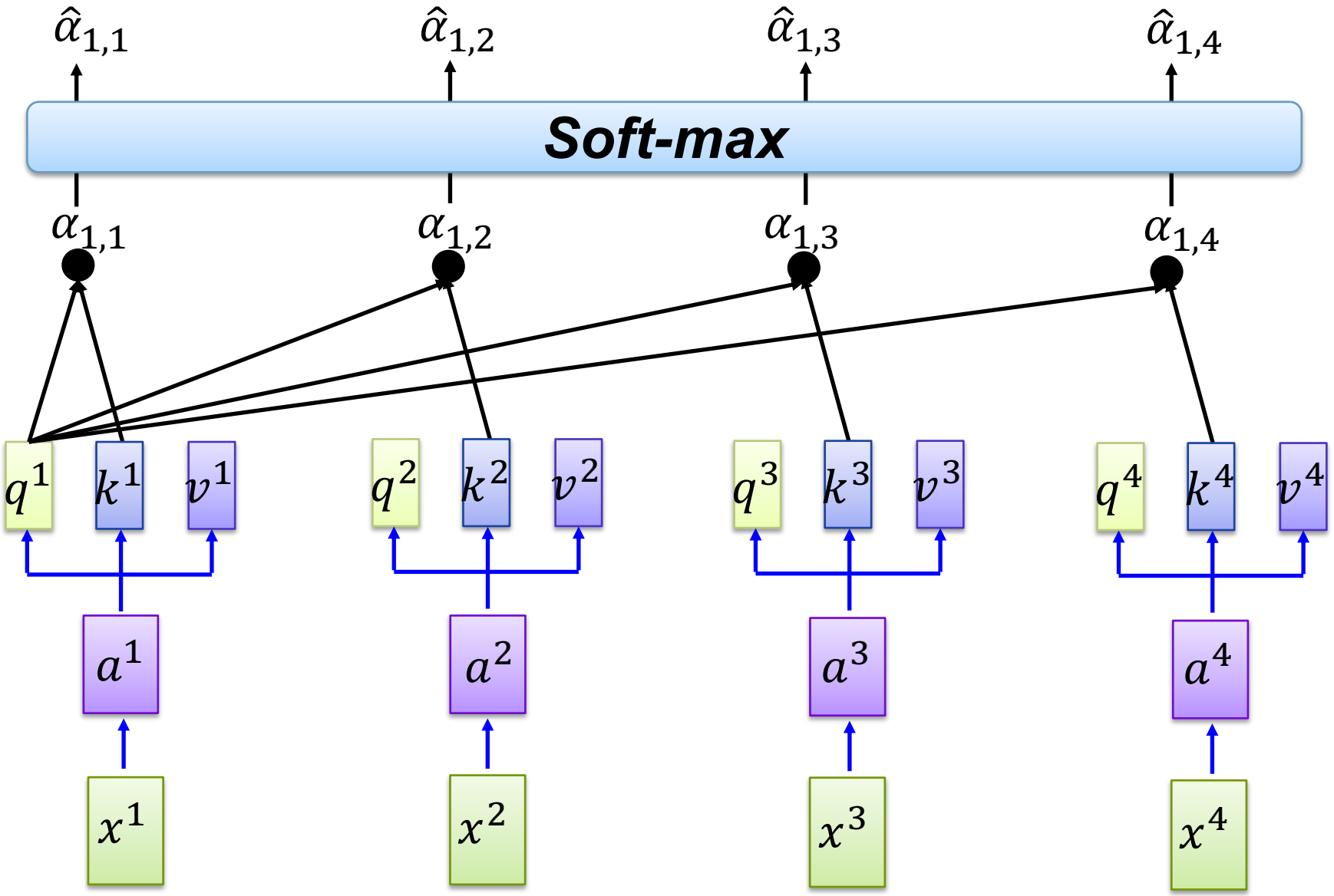
d is the dim of q and k

Scaled Dot-Product Attention: $\alpha_{1,i} = \underbrace{q^1 \cdot k^i}_{\textbf{dot product}} / \sqrt{d}$



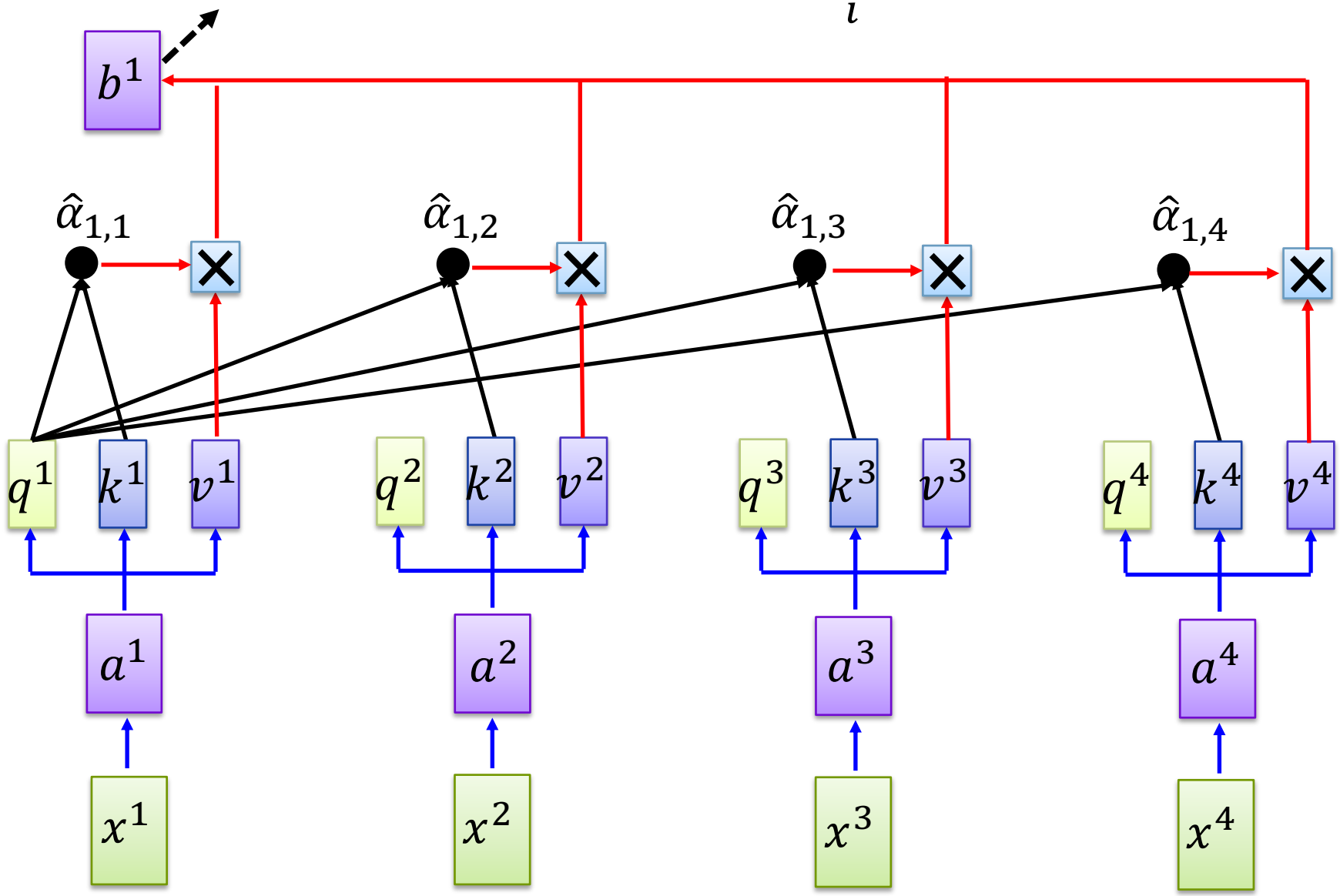
Self-attention

$$\hat{\alpha}_{1,i} = exp(\alpha_{1,i}) / \sum_j exp(\alpha_{1,j})$$

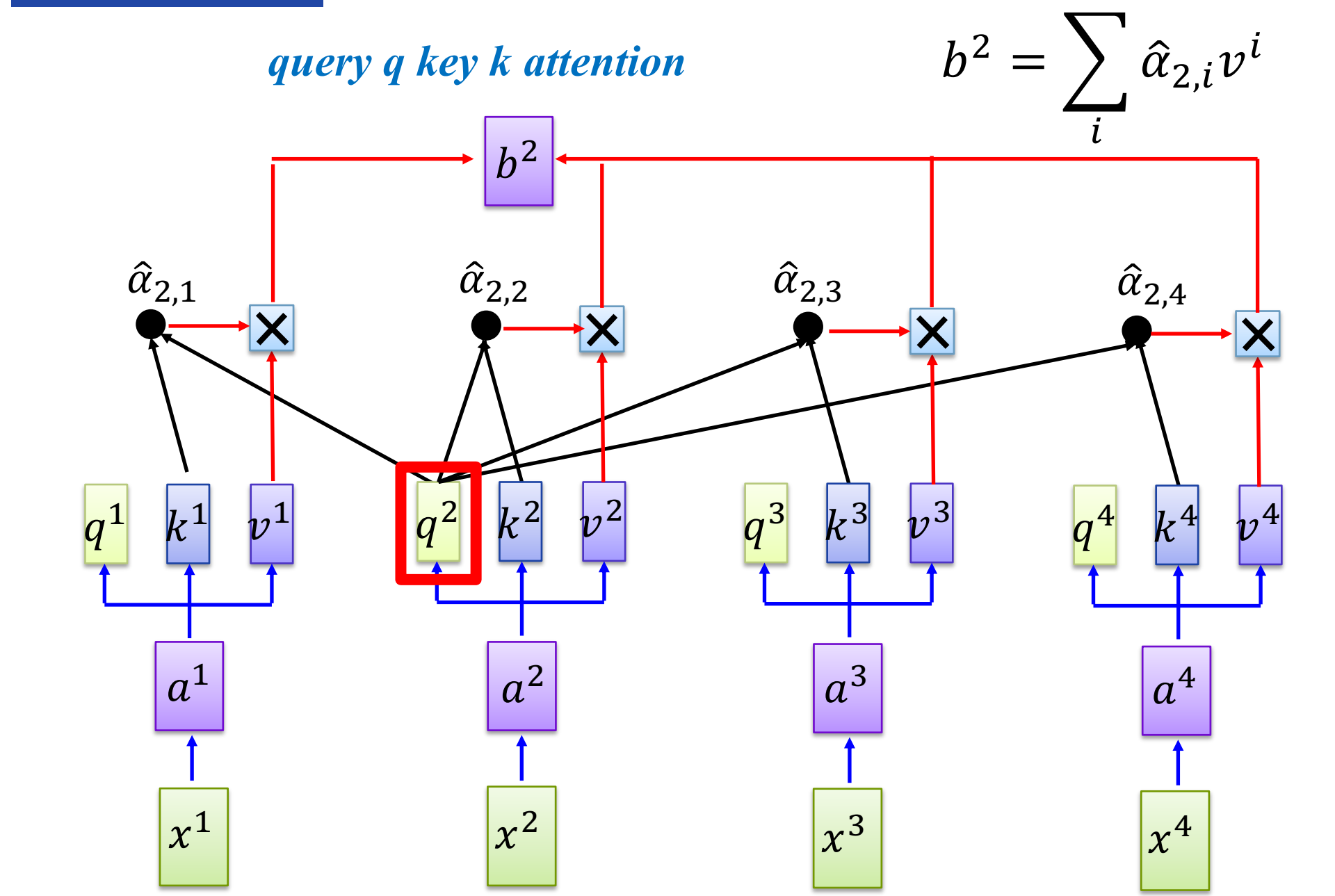


Self-attention

$$b^1 = \sum_i \hat{\alpha}_{1,i} v^i$$

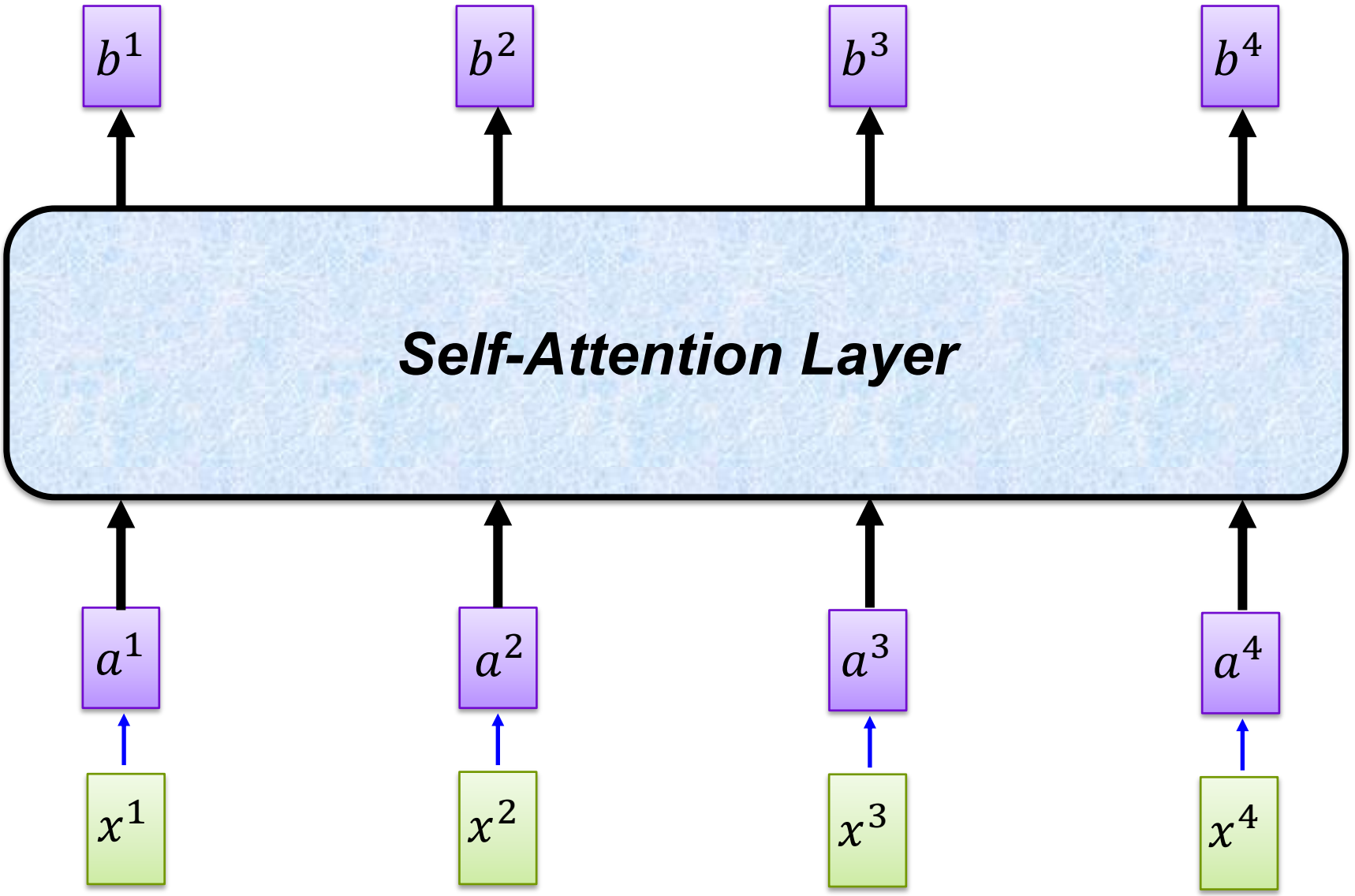


Self-attention



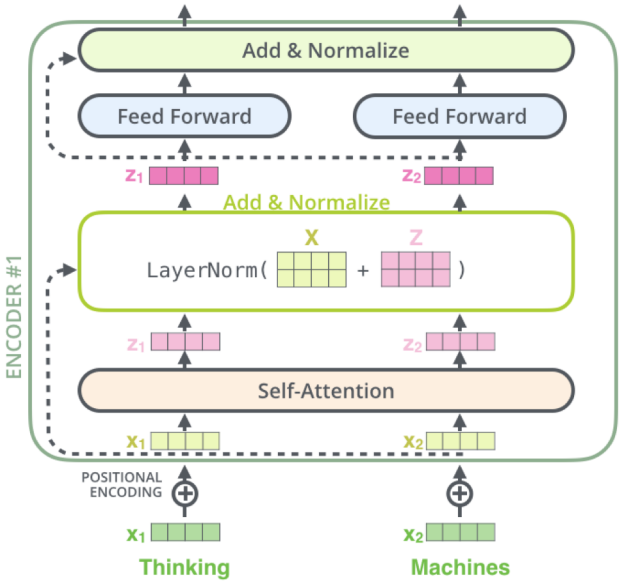
Self-attention

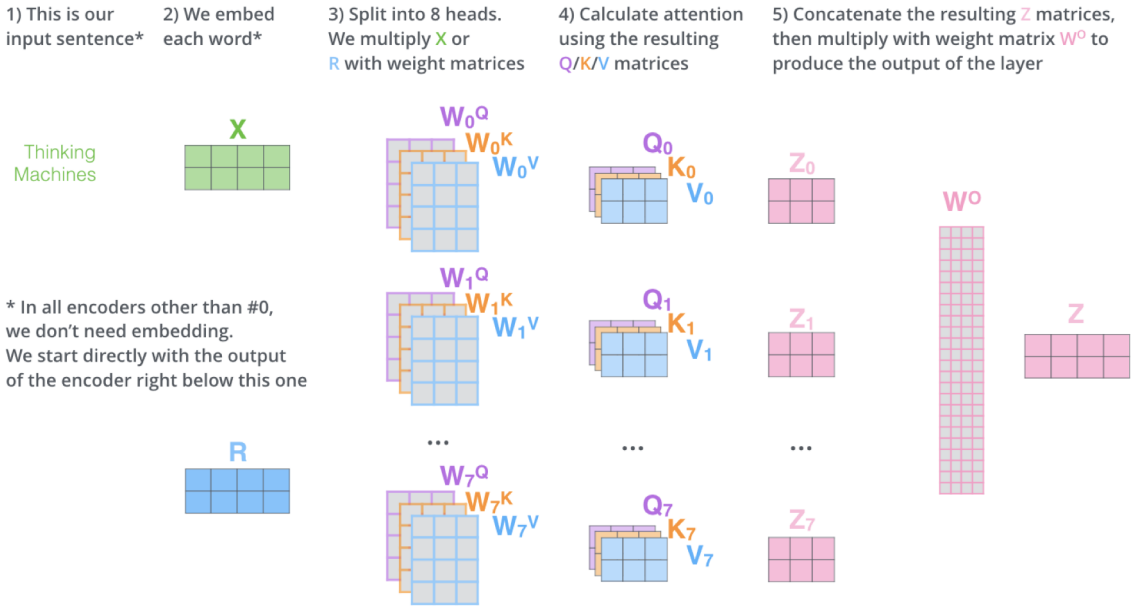
b^1, b^2, b^3, b^4 ***can be parallelly computed.***



Transformeurs (*Tranformers*)

Attention multi-têtes (Multi-head attention)





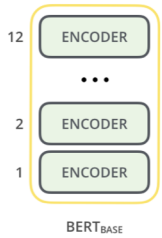
© alammar.github.io/

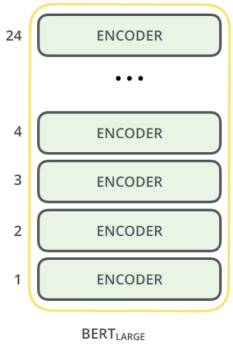
Le modèle BERT

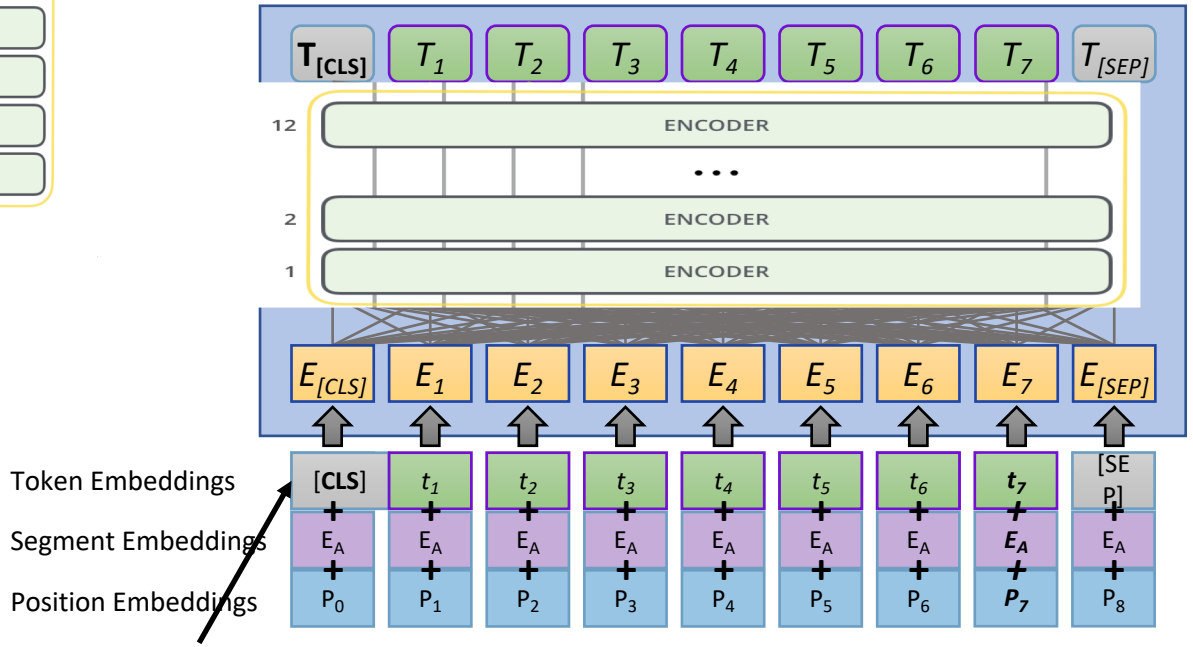
- BERT is an attention mechanism that learns contextual relations between words (or sub-words) in a text (apprend **des modèles de langues**)
- BERT = un Encodeur (pas de décodeur), développé par Google
- BERT (and BERT like models RoBERT, AlBERT, …) est la nouvelle “star” du TAL (traitement automatique des langues) et plus généralement des tâches exploitant le texte (recherche d’information, traduction automatique, système questions réponses, génération automatique de textes, …)

Illustration BERT

- En pratique, Deux BERT ont été proposés
  - BASE : 12 encoders, , 12 Attention, Dim E/S (768
  - LARGE : 24 encoders, 16 attention heads, dimension vecteur d'E/S 1024



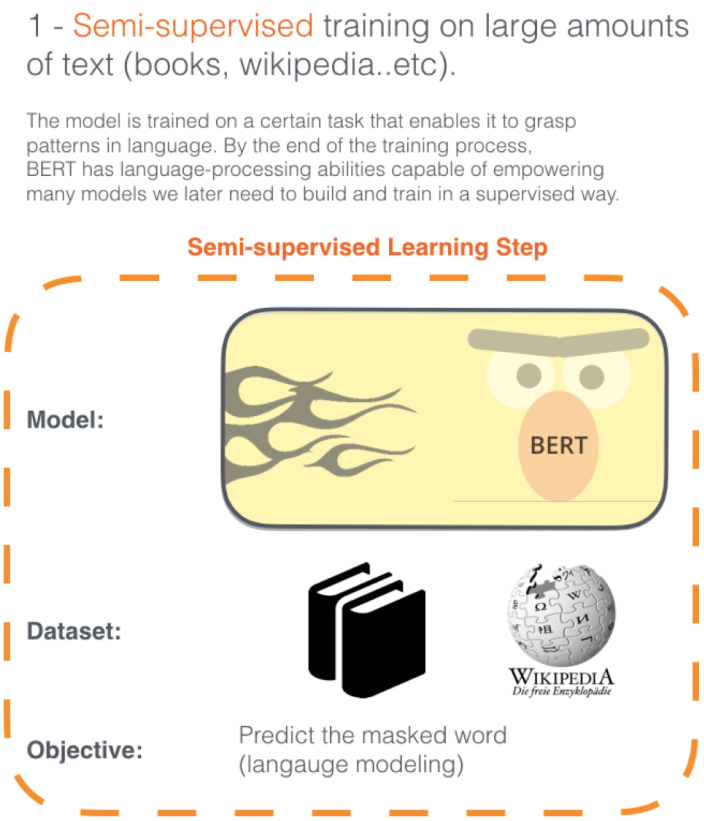


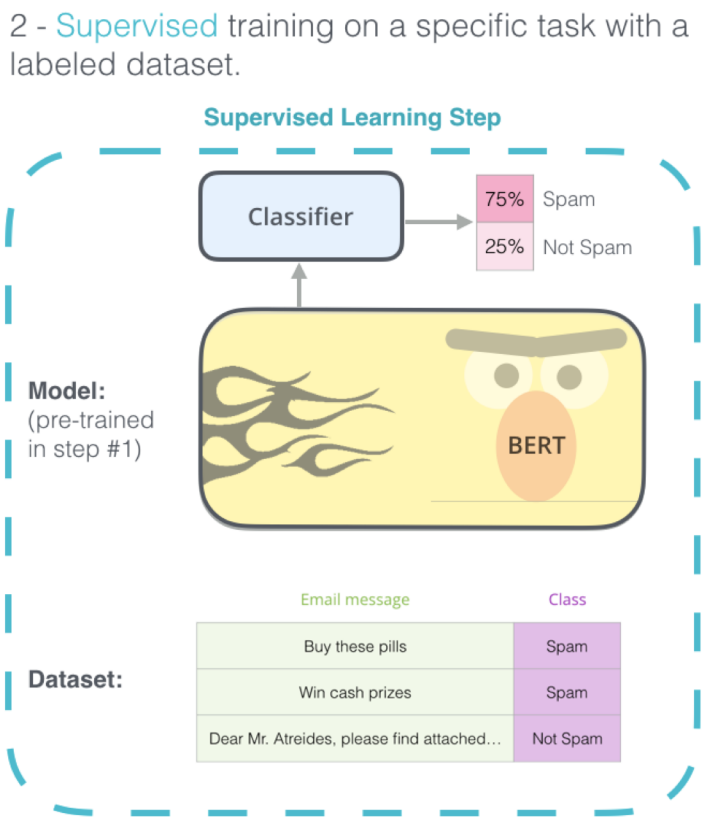


© alammar.github.io/ Le premier ***token*** d'entrée est un jeton spécial [CLS].

BERT illustration

- En pratique
  - Etape1: on apprend un modèle de langue
  - Etape 2 : on exploite le modèle sur une tâche supervisée (finetuning)

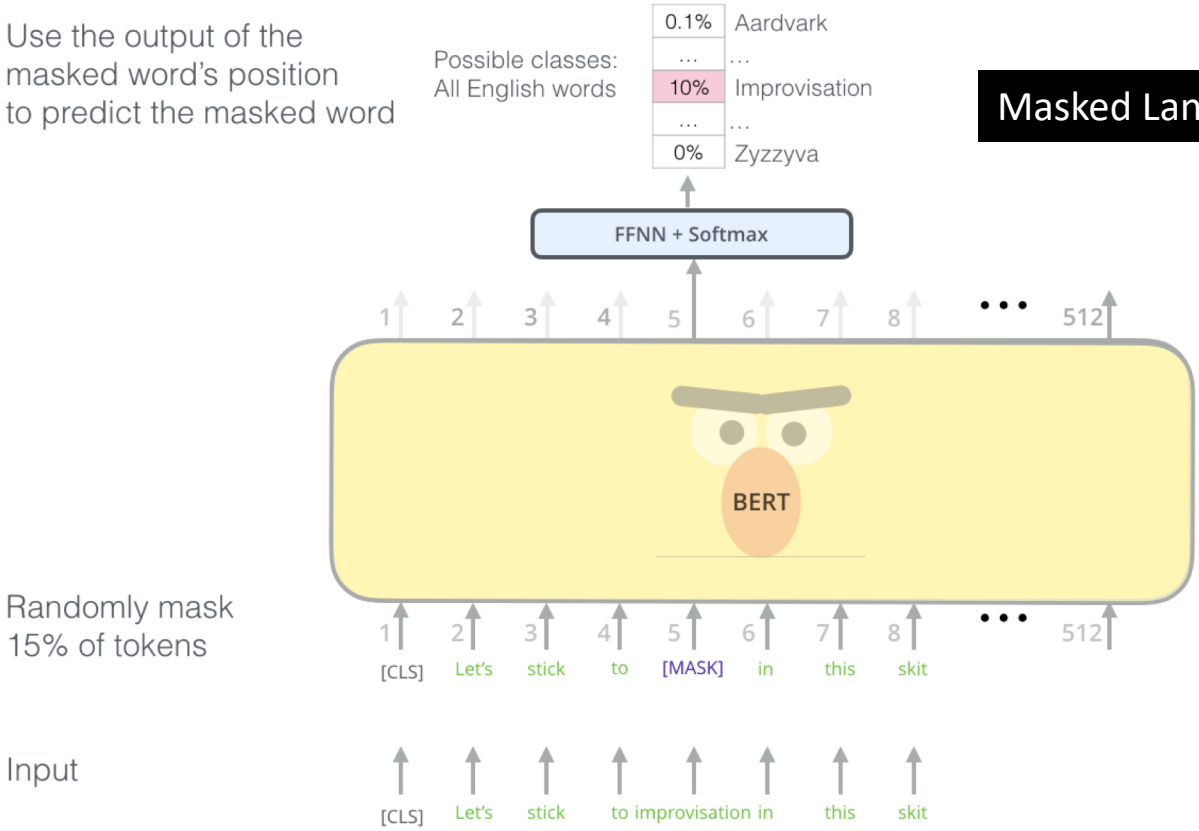




BERT : Phase d'apprentissage

Apprentissage d'un modèle de langue (dans le cadre de BERT),

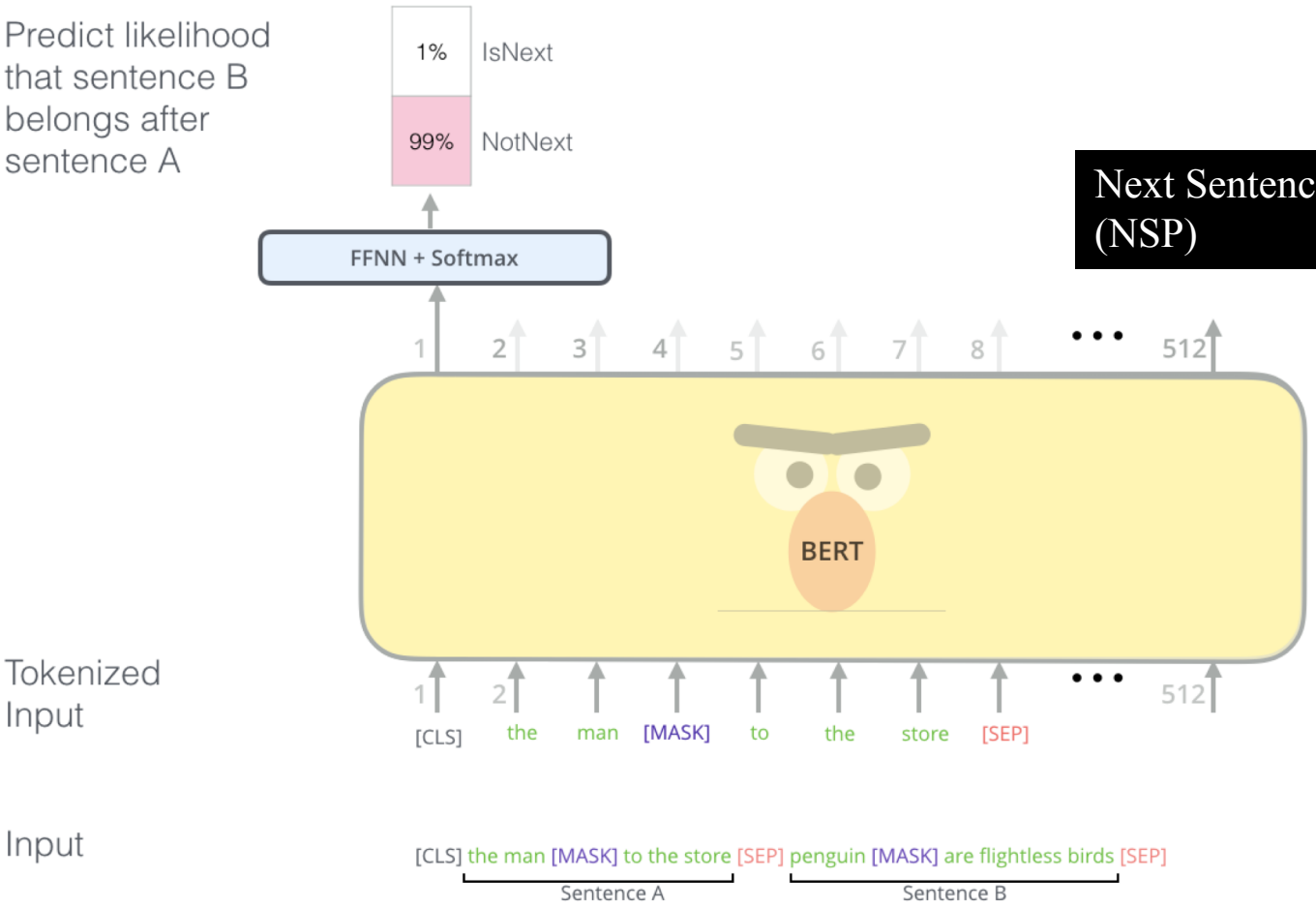
2 tâches : masked word prediction (Masked Language Model) et Next Setence Prediction



Masked Language Model

© *alammar.github.io/*

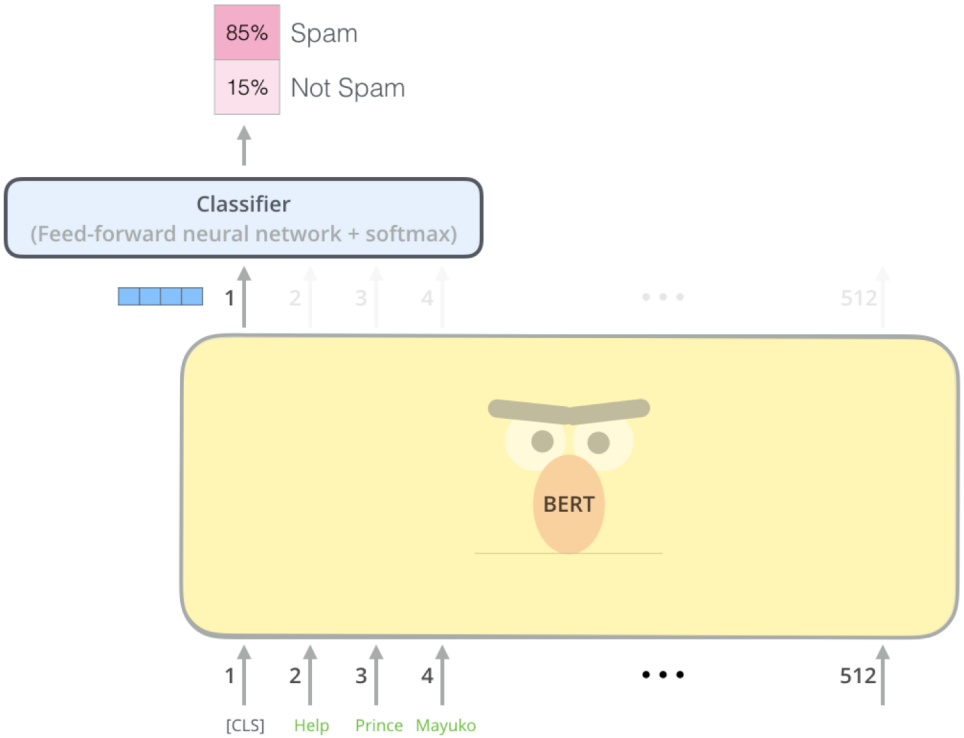
BERT : Phase d'apprentissage



Next Sentence prediction (NSP)

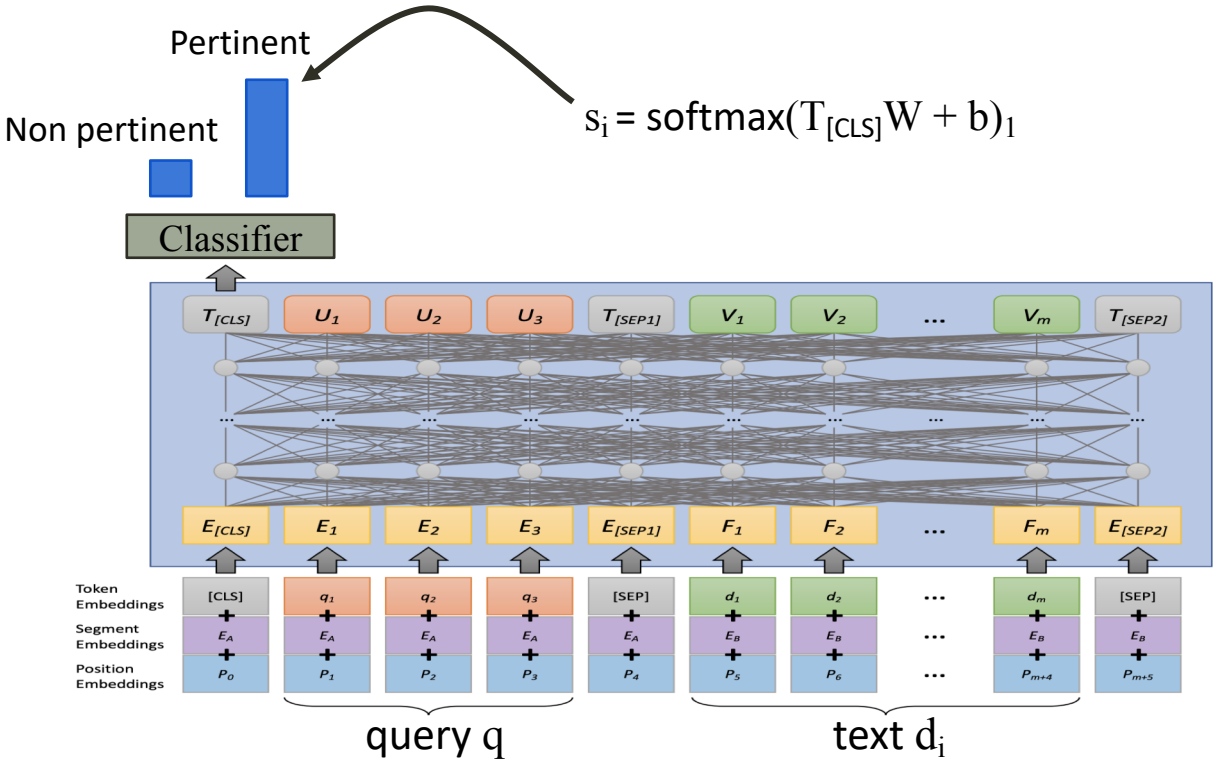
BERT : Phase d'affinage (fine-tuning)

- Affinage (utilisation de BERT pour entraîner sa propre tâche)



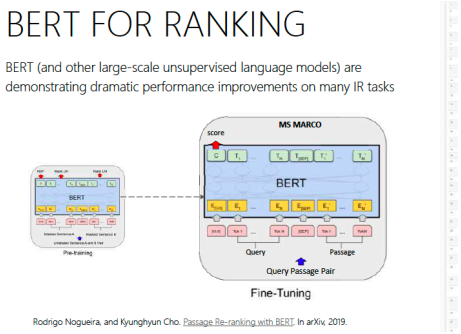
© *alammar.github.io/*

- Affinage (Fine-tuning)
  - Utiliser un BERT pré-entrainé (prendre un checkpoint de chez Google ou de hugginface)
  - Prendre une collection d'apprentissage : Requêtes-documents Pertinents et non pertinents.
  - Fournir en entrée de BERT requête+"partie du doc" (Pb pas plus de 512 tokens)



BERT en action en RI

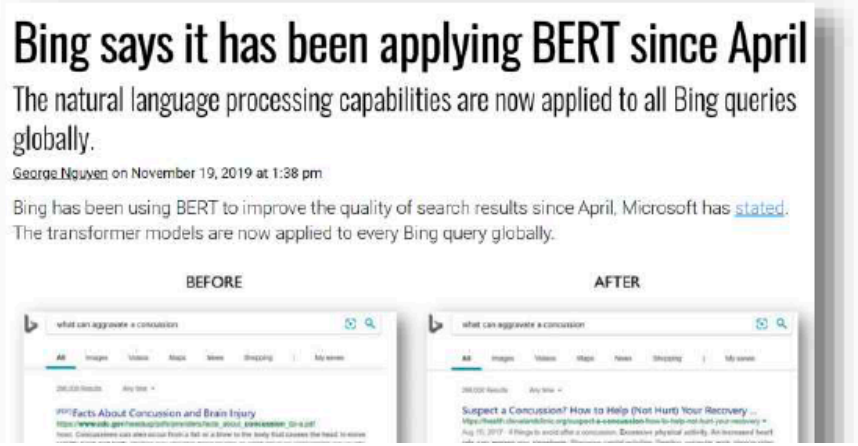
- Sélectionner une liste de documents (100, 1000) en utilisant un modèle classique (BM25)
- Faire du fine-tuning
  - Utiliser un BERT affine *"finetuné" (c'est mieux)*
  - Découper le document en petits morceaux pour faire au total (requête+document+SEP+CLS pas plus 512 mots)
  - Fournir en entrée de BERT requête+"partie du doc"
  - Bert renvoie un score de pertinence



BERT en action en RI

BERT FOR RANKING

Impact across both academia and industry



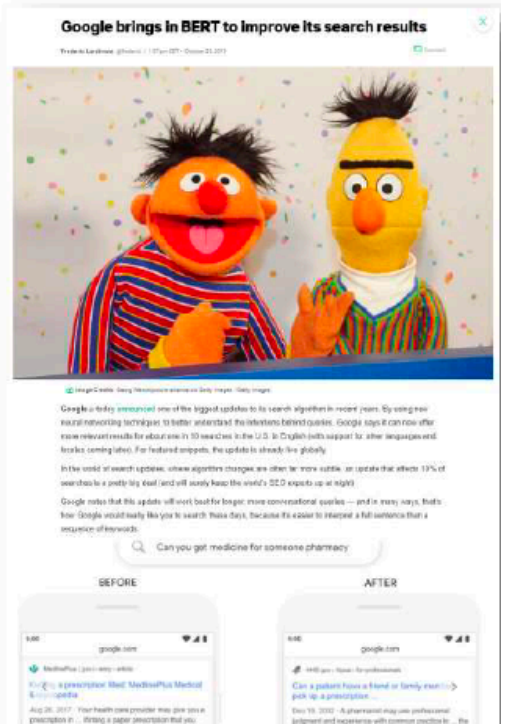
Bing says it has been applying BERT since April

The natural language processing capabilities are now applied to all Bing queries globally.

George Nguyen on November 19, 2019 at 1:38 pm

Bing has been using BERT to improve the quality of search results since April. Microsoft has stated. The transformer models are now applied to every Bing query globally.

BEFORE AFTER



Google brings in BERT to improve its search results

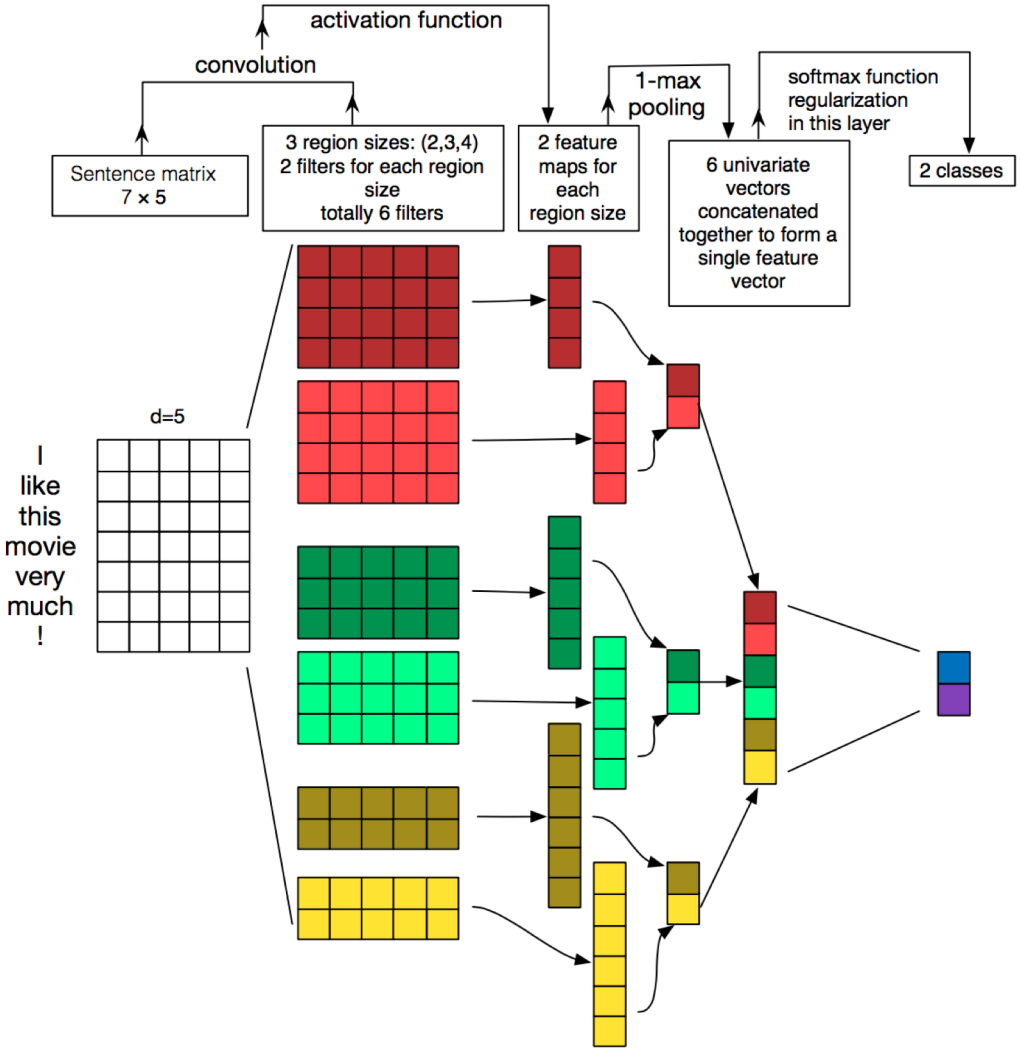
Can you get medicine for someone pharmacy

BEFORE AFTER

- Utiliser le notebook sur Google Colab : BERT FineTuning with Cloud TPUs.

- Panoplie de modèles préentrainés : https://huggingface.co/

Outils pour faire du Deep learning



Zhang and Wallace, 2015; Sentence classication