Modèles neuronaux pour la modélisation statistique de la langue

Fethi BOUGARES

Maître de conférences
Laboratoire d'Informatique de l'Université du Maine

2015 - 2016

Sommaire

Modélisation de langage

- Caractériser la qualité des énoncés en langue naturelle
- Évaluer la qualité de ses énoncés
- Rôle est fondamentale dans de nombreux cadres d'application :
  - Reconnaissance automatique de la parole
  - Traduction automatique
  - Extraction et la recherche d'information

Sommaire

Modélisation de langage $n-gramme$

- modélisation actuellement état de l'art
- prédit un mot en fonction des $n-1$ mots précédents

1. Chaque mot traité comme un symbole discret qui n'a pas de relation avec les autres
2. Ordre est limité à $n = 4$ ou 5 : caractère éparse de la langue
3. Construction : dénombrement de successions de mots, effectué sur des données d'entrainement

Modélisation de langage $n-gramme$

Chaque probabilité du modèle $n-gram$ est estimée grâce à la fréquence relative calculée sur un corpus d'apprentissage en utilisant plusieurs fonctions d'interpolation.
Le modèle $n-gram$ a des inconvénients :

- inefficace avec grand n ($n > 5$)
- La structure inhérente au langage est ignorée (la similarité entre les mots) :

  Si on observe les phases suivantes :
  L'étudiant commence son stage.
  Le thésard démarre sa thèse
  Que dire de la phrase :
  Le thésard commence sa thèse.

Modélisation de langage $n-gramme$

On ne peut pas prédire efficacement sa probabilité car il n'y a aucun lien entre deux mots :

- commence
- démarre

Sommaire

Modélisation neuronale $n-gramme$

Introduit par l'article de (Bengio et al., 2003) : les réseaux neuronaux pour la modélisation du langage

$\Rightarrow$ Idée principale est d'associer (ou de projeter) chaque mot à un vecteur dans un espace continu

$\Rightarrow$ utiliser le réseau neuronal pour apprendre la distribution comme une fonction des vecteurs.

Définition mathématique

Un réseau de neurones (avec une couche cachée) est simplement une fonction ayant la forme suivante :

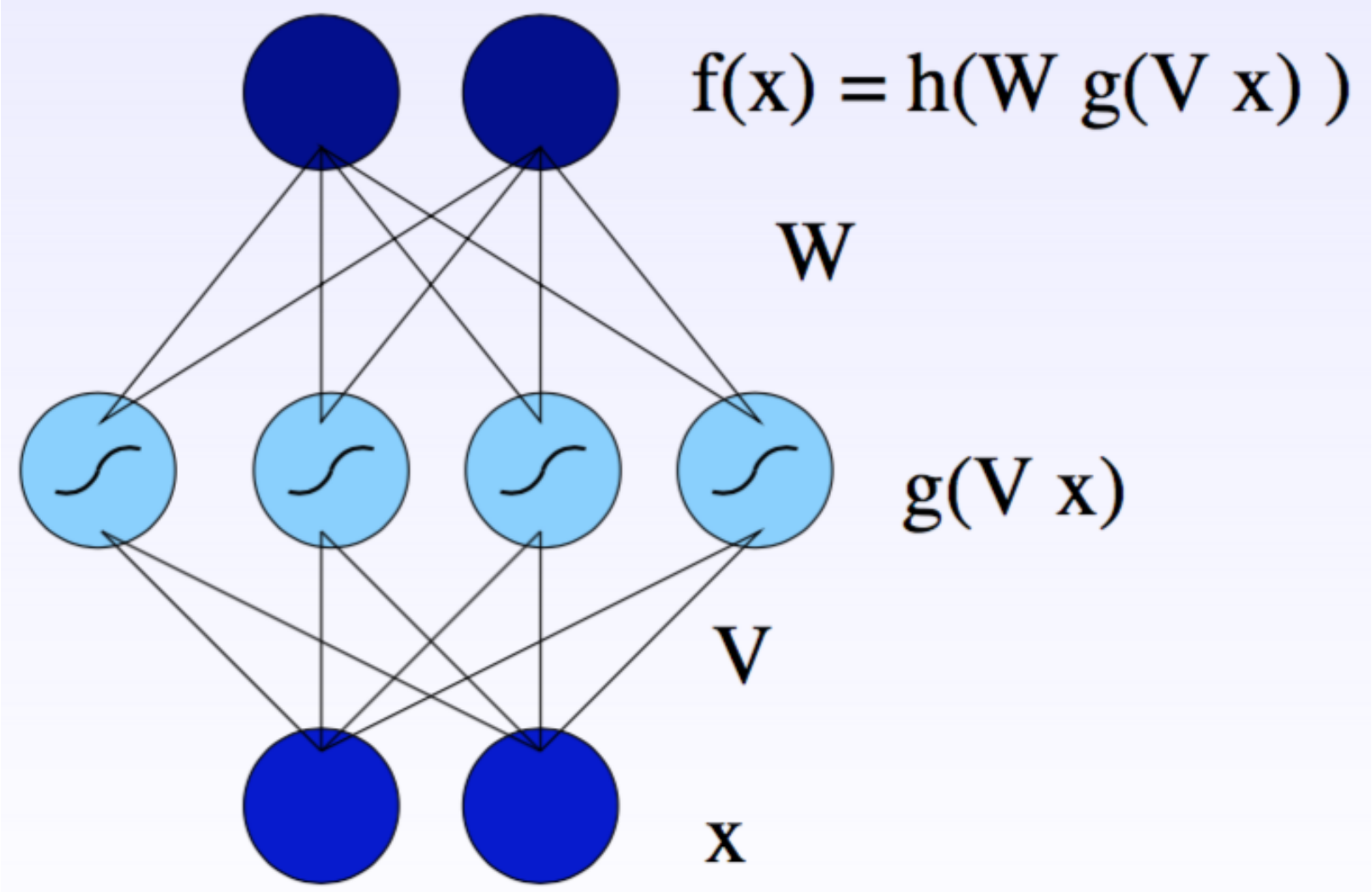
$$f(x) = h(o(x)) \quad \text{avec} \quad o_i = \sum_j W_{ij} a_j(x)$$

$$a_j(x) = g(\sum_k V_{ij} x_k)$$

$g(.)$ et $h(.)$ sont des fonctions d'activation

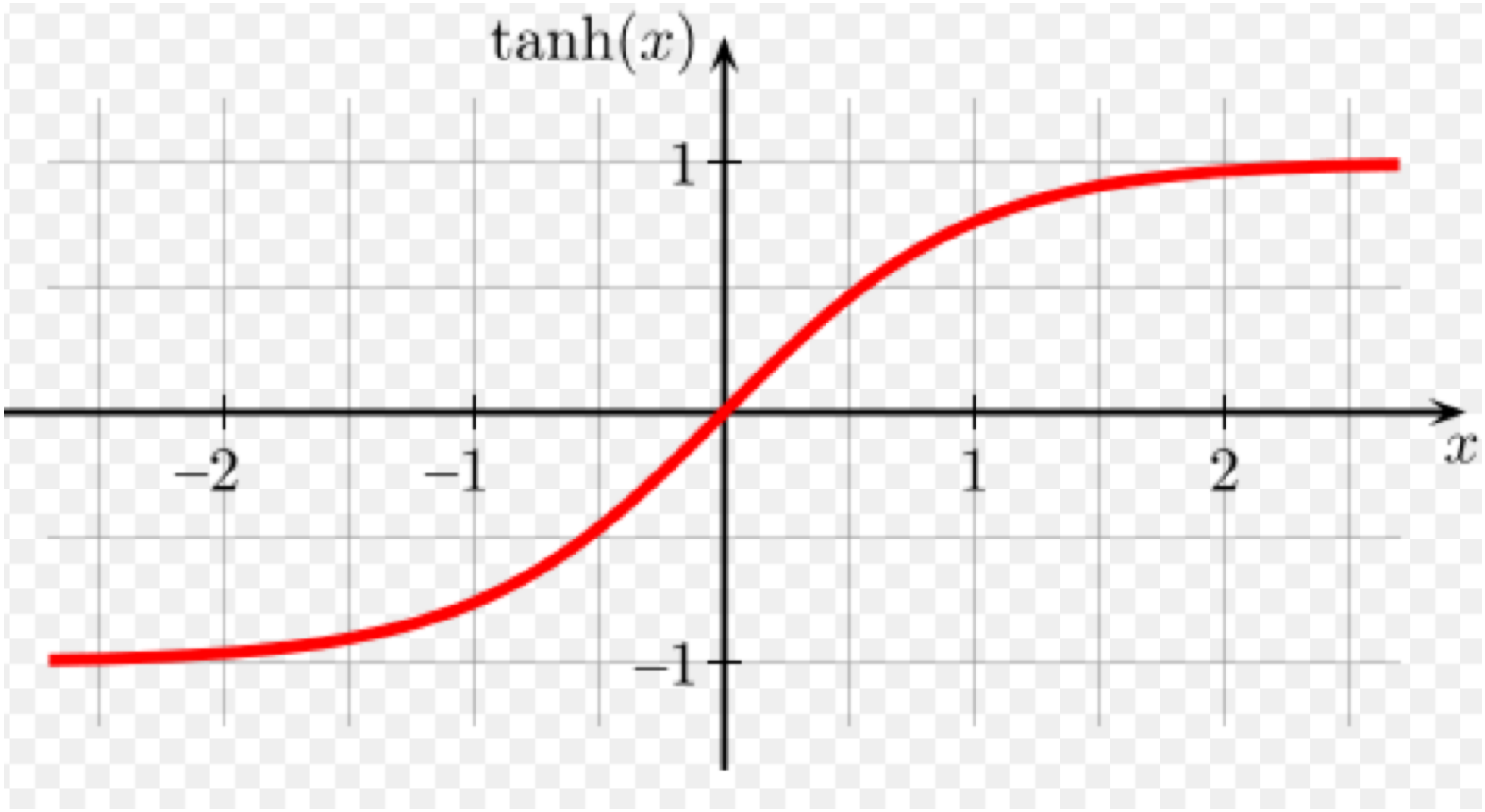
$\Rightarrow$ Les réseaux de neurones peuvent approximer la plupart des fonctions, donc peuvent résoudre la plupart des tâches
$\Rightarrow$ Ce qui manque : un moyen d'obtenir les matrices W et V
$\Rightarrow$ Un algorithme d'apprentissage pour accomplir cette tâche

Le réseau



Fonctions d'activation

Couche cachée :



Fonctions d'activation

Couche de sortie : $Softmax$

$$h(o_i) = \frac{exp(o_i)}{\sum_j exp(o_j)}$$

$\Rightarrow$ On pourra interpréter la sortie $y_t = f(x_t)$ comme étant la probabilité que $y(t)$ soit associé à x_t selon notre modèle

Algorithme d'apprentissage

- SGD : Descente de Gradient Stochastique
- Données d'apprentissage :
  Un ensemble de paires : $\mathrm{T} = \{(x_t, y_t)\}_{t=1}^{n}$
- Une fonction objective : un coût à optimiser $\Rightarrow$ La log-vraisemblance négative des données

$$C(x_t, y_t, f(.)) = -logP(y_t|x_t, f(.))$$

Algorithme d'apprentissage : Rétropropagation du gradient

Étapes :

1. Initialiser W et V
2. De t =1 à $|T|$
   - 2.1 calculer $C(x_t, y_t, f(.))$
   - 2.2 mettre à jour $W_{ij} \leftarrow W_{ij} - \lambda \frac{\partial}{\partial W_{ij}} C(x_t, y_t, f(.)) \forall i, j$
   - 2.3 mettre à jour $V_{jk} \leftarrow V_{jk} - \lambda \frac{\partial}{\partial V_{jk}} C(x_t, y_t, f(.)) \forall j, k$
3. si le critère d'arrêt n'est pas satisfait, retourner à 2

Critère d'arrêt : la progression du coût sur un ensemble de validation. On arrête l'apprentissage lorsque ce coût augmentera.

Espace continue : représentation

La représentation distribuée d'un objet (e.g. mot) est simplement un vecteur de valeur réelle, de taille fixe D. Exemple :

Mots	Représentation
”le”	[0.6762 -0.9607 0.3626 -0.2410 0.6636]
”la”	[0.6859 -0.9266 0.3777 -0.2140 0.6711]
”.”	[-0.0069 0.7995 0.6433 0.2898 0.6359]
”,”	[0.0295 0.8150 0.6852 0.3182 0.6545]
chien	[0.5896 0.9137 0.0452 0.7603 -0.6541]
chat	[0.5965 0.9143 0.0899 0.7702 -0.6392]
....	...

Espace continue : représentation

Il existe plus d'un algorithme pour apprendre les représentations distribuées des mots d'un vocabulaire

- Latent Semantic Analysis
- Semidefinite Embedding
- **Les réseaux de neurones**

Modélisation de langage

La tâche correspond à :

$$W_{t-N+1}^{t-1} \rightarrow W_t$$

$\Rightarrow$ Estimer W_t étant donnée W_{t-N+1}^{t-1}

Évaluation : Perplexité

Modélisation de langage : Perplexité (PPL)

- Étant donnée un corpus de test de m phrases : $s_1, s_2, ..., s_m$
- Chaque phrase contient n mots : $w_1, w_2, ..., w_n$
- Pour chaque phrase s on pourra calculer $p(s_i)$ selon notre LM

$$\prod_{i=1}^{m} p(s_i) \qquad (1)$$

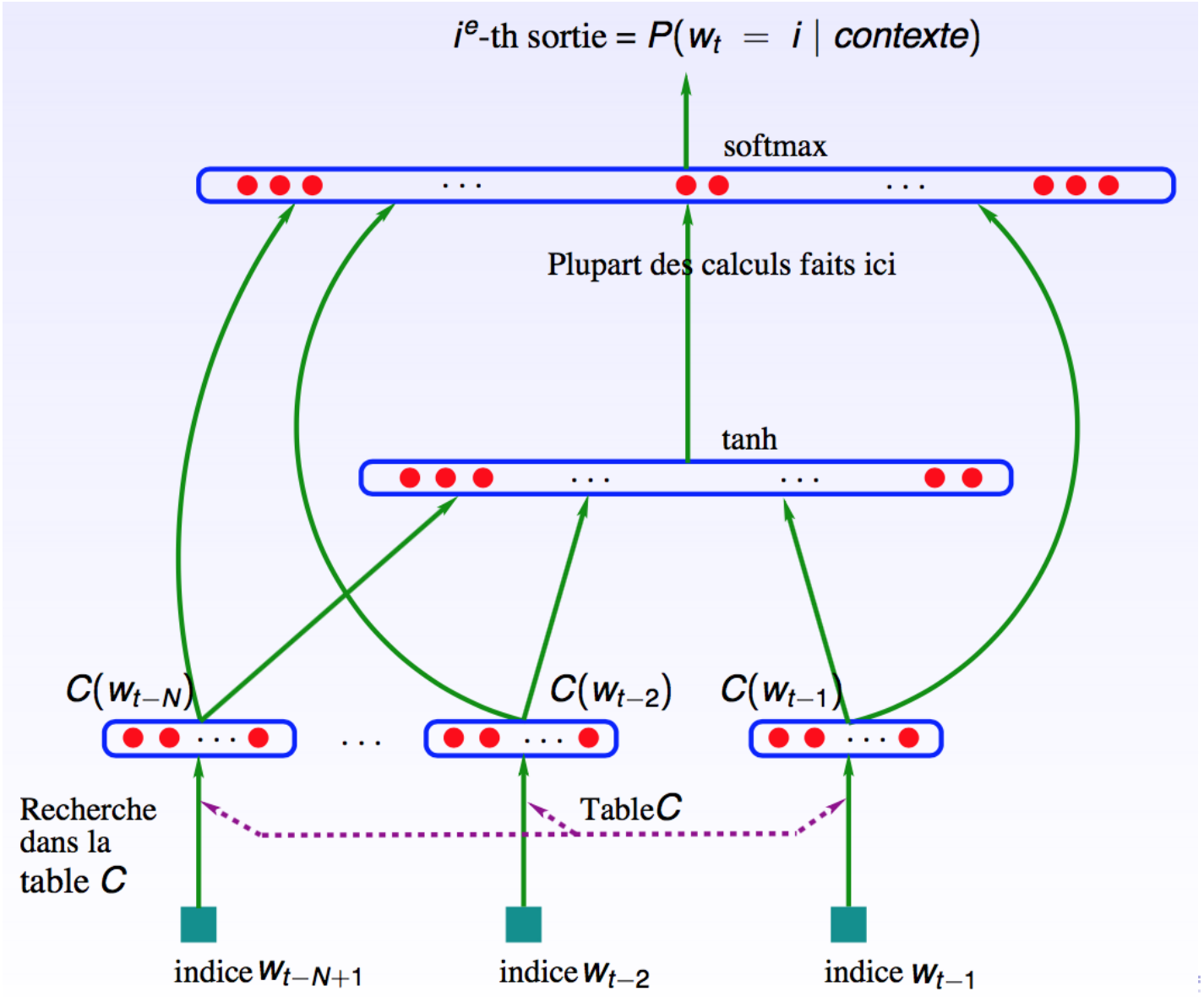
Une probabilité donc : plus c'est haut, mieux c'est.

1. **Log prob AVG** :

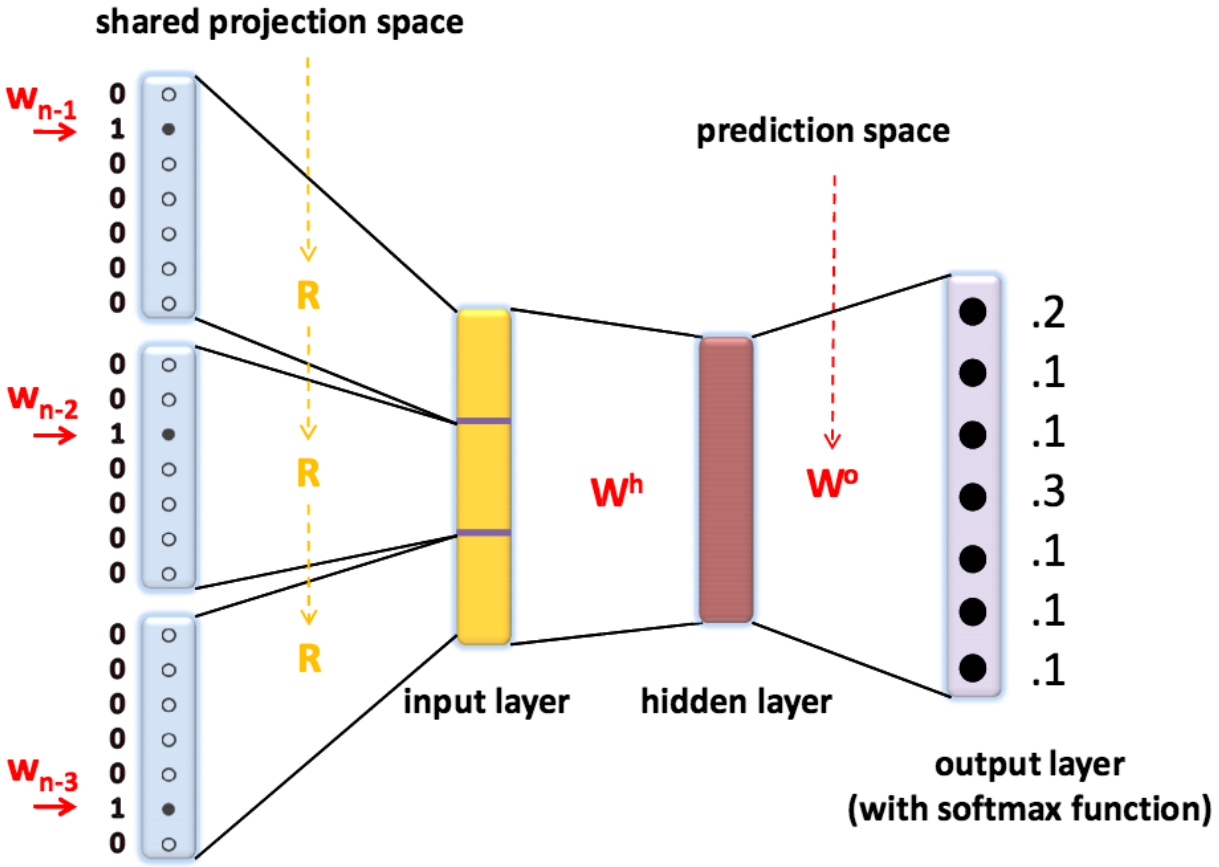
$$\frac{1}{M} log \prod_{i=1}^{m} p(s_i) = \frac{1}{M} \sum_{i=1}^{m} p(s_i)$$

2. **PPL** : 2^{-l} avec $l = \frac{1}{M} \sum_{i=1}^{m} p(s_i)$ plus c'est bas, mieux c'est.

Modélisation de langage



Modélisation de langage



Modélisation de langage

- Vocabulaire : Taille de la dernière couche
- Chaque neurone de sortie correspond à la probabilité d'un mot
- Le calcul se concentre à la couche de sortie
- Le temps de calcul est donc linéaire par rapport à la taille du vocabulaire
- Il est possible d'utiliser une *shortlist* , i.e, seuls les mots les plus fréquents sont prédits

Architecture et méta-paraméters

Comment choisir l'architecture et fixer les méta-paramètre ?

- Combien de couche cachée ?
- Comment initialiser les matrices de poids ?
- Comment modifier le pas d'apprentissage *learningrate* ?
- Quelle taille de n-gram ?
- Comment fixé taille de *batch* ?

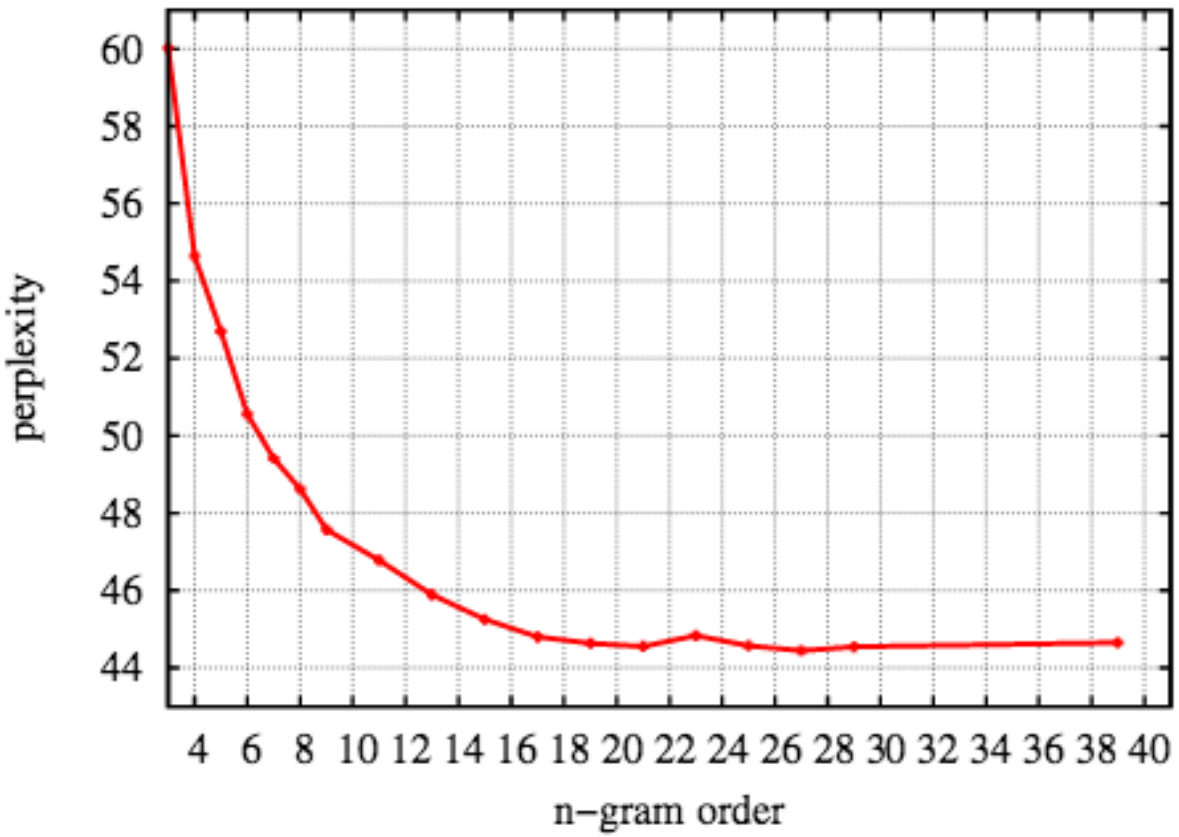
Résultats

Données d'apprentissage :

Training data	genre	words	subset
Parallel	informal	7.1M	7.1M
(target side)	mixed	178M	19.7M
Monolingual	news	4 800M	173M
	informal	666M	33.3M
Total		**5 651M**	**233M**

Résultats

Perplexité en fonction de l'ordre de n-gram :



Sommaire

Intégration d'un NN-LM

1. première passe : produire un treillis d'hypothèse
2. deuxième passe : Mise à jour des probabilités du treillis de façon efficace

$\Rightarrow$ Très rapide 0.05 x RT

Résultats

LM training data	Back-off LM	Neural LM alone	Hybrid LM
	84 M	84 M	84 M
Perplexity	122.6	117.3	107.8
Word error rate	23.5%	23.2%	22.6%

Traduction automatique statistique

Traduire une phrase source s consiste à trouver la phrase cible c^* qui est la traduction la plus probable de s.

$$\begin{aligned}\hat{c} &= \underset{c}{\text{argmax}}\, P(c|s) \\ &= \underset{c}{\text{argmax}}\, P(s|c)P(c)\end{aligned}$$

$P(s|c)$: Modèle de traduction
$P(c)$: Modèle de langage
argmax : Algorithme de décodage

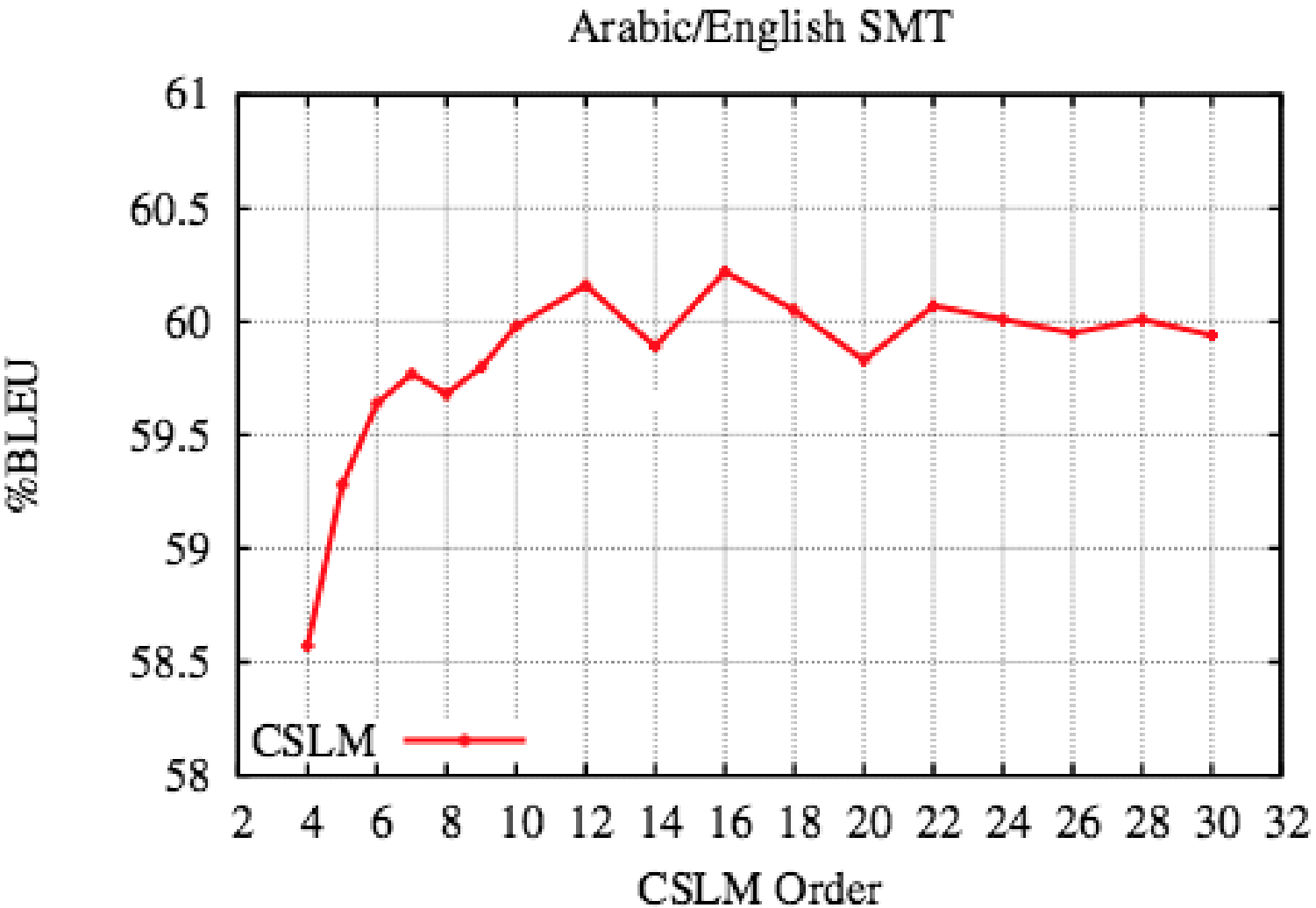
Intégration d'un NN-LM

1. première passe : produire une liste de n-best d'hypothèse
2. deuxième passe : calculer d'un score NNLM et ré-ordonner les phrases

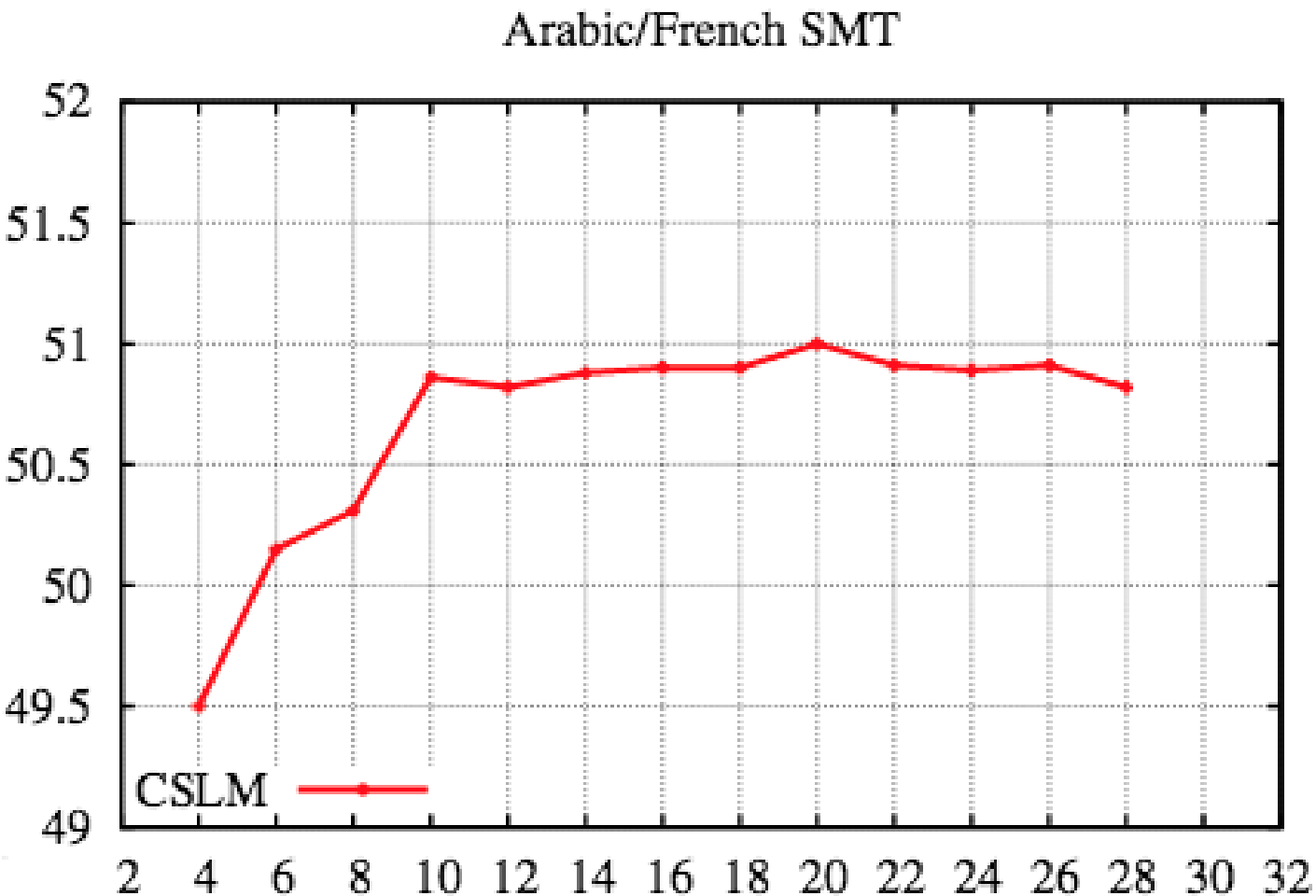
Intégration d'un NN-LM

1. première passe : produire une liste de n-best d'hypothèse
2. deuxième passe : calculer d'un score NNLM et ré-ordonner les phrases

Résultats


Arabic/English SMT
%BLEU
61
60.5
60
59.5
59
58.5
58
CSLM
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32
CSLM Order

Résultats


Arabic/French SMT
52
51.5
51
50.5
50
49.5
49
CSLM
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32

Et après ?

- Compréhension de la parole
- Recherche d'Information
- Reconnaissance de la parole
- Traduction avec un réseau de neurone
- ....

Skype Translator (demo) ?

Questions ?