

Part Of Speech Tagger

TRAITEMENT AUTOMATIQUE DE LA LANGUE

NEYRAUD An Toàn

Polytech Paris-Saclay

Table des matières

[1. Introduction 3](#_Toc96788407)

[2. Présentation du modèle neuronal 4](#_Toc96788408)

[2.1. Présentation de l’architecture neuronale du POS tagger 4](#_Toc96788409)

[2.2. Les deux modèles : 5](#_Toc96788410)

[2.3. Apprentissage du modèle neuronal 5](#_Toc96788411)

[3. Evaluation du POS Tagger 6](#_Toc96788412)

[3.1. Description des corpus 6](#_Toc96788413)

[3.2. Description des métriques d’évaluations 6](#_Toc96788414)

[3.3. Résultats 6](#_Toc96788415)

[4. Points forts 7](#_Toc96788416)

[5. Limites du modèle 7](#_Toc96788417)

[6. Difficultés rencontrées 8](#_Toc96788418)

[7. Organisation 8](#_Toc96788419)

[8. Conclusion 8](#_Toc96788420)

[9. Annexe 9](#_Toc96788421)

# Introduction

Dans le cadre de ma formation au sein de l’école Polytech Paris-Saclay, j’ai eu l’occasion de recevoir l’enseignement de Mr SEMMAR Nasredine (Research Scientist chez CEA) sur ce qu’était la NLP (Natural language processing). Durant ce dernier, j’ai pu réaliser un projet de fin de module sur une période d’environ 2 semaines.

Ce projet a pour objectif l’implémentation d’un désambiguïseur morpho-syntaxique. C’est-à-dire la conception d’un POS Tagger, c’est un logiciel qui lit du texte dans une langue et attribue des étiquettes à chaque mot (ou groupe de mots suivant le POS Tagger), tels que le nom, le verbe, l’adjectif, etc…

Pour ce faire, j’ai réalisé un modèle de Lachine Learning dans le langage Python, avec la librairie PyTorch. Pour le modèle, il nous était conseillé d’utiliser un réseau de neurones récurrent, de type LSTM (Long short-term memory).

# Présentation du modèle neuronal

## Présentation de l’architecture neuronale du POS tagger

Pour la réalisation du modèle, j’ai utilisé le Word Embedding (Plongement de mot en Français) pour favoriser l’apprentissage du modèle. Concrètement, c’est de la vectorisation de mots, c’est-à-dire que cette technique permet de représenter chaque mot d'un dictionnaire par un vecteur de nombres réels. Ce qui veut dire que deux mots étant assez proches dans leurs significations auront des vecteurs assez proches aussi.

Pour ce qui est du réseau de neurone, j’ai utilisé un réseau de neurones récurrents qui est un LSTM (Long short-term memory). Ce dernier est divisé en couche LSTM et dans un chaque couche, nous avons trois porte, la porte d’oubli (cf. tracé rouge figure 1), la porte d’entrée (cf. tracé vert figure 1) et la poste de sortie (cf. tracé bleu figure 1).

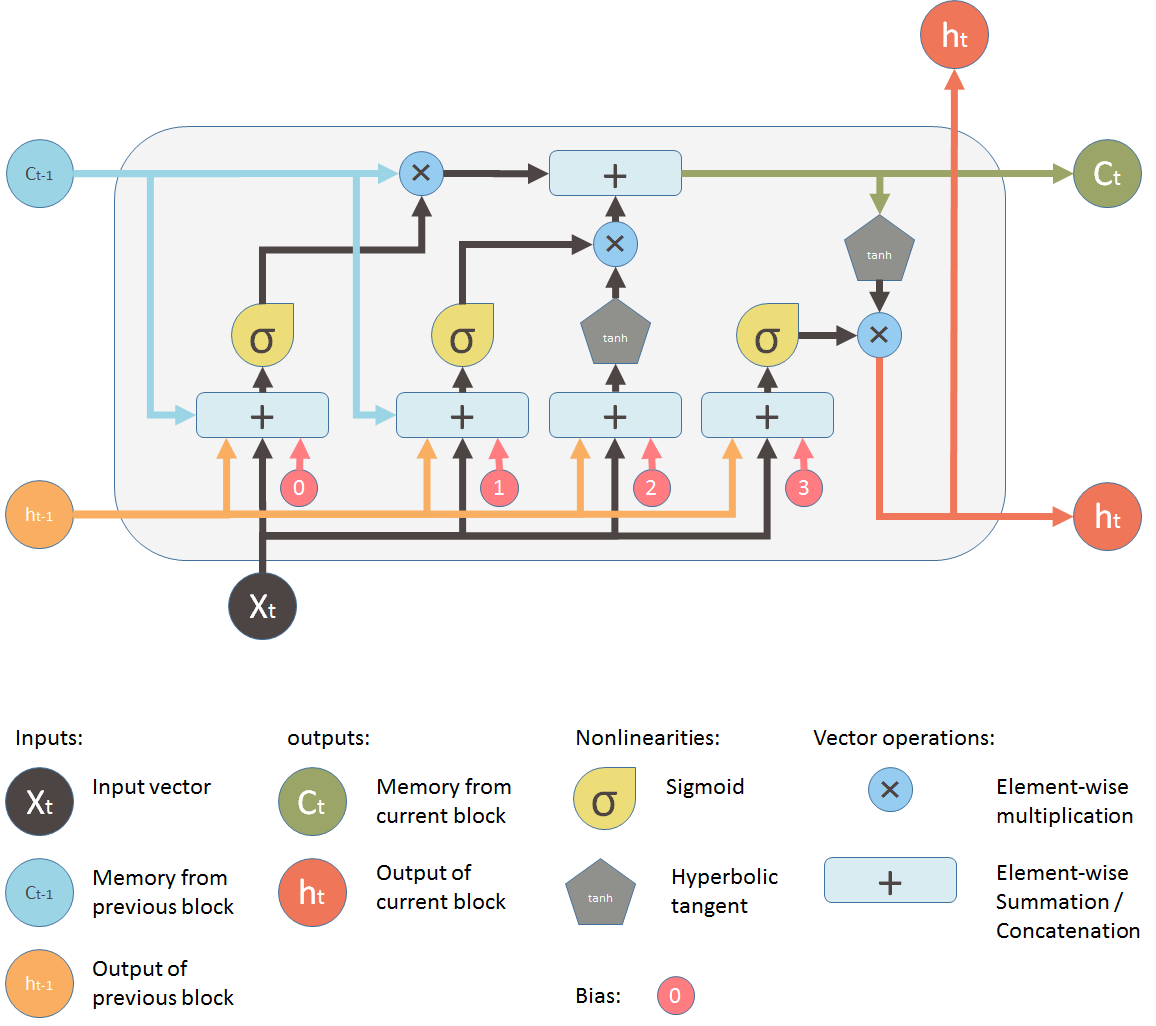


Figure 1 : Schéma d'une couche LSTM

Rôles des portes :

* La porte d’oubli se charge uniquement de filtrer les anciennes valeurs contenues dans la cellule mémoire précédente.
* La porte d’entrée va, parmi les nouvelles entrées, laisser passer les informations qui sont pertinentes. Ces valeurs-là seront stockées dans la cellule mémoire et auront une importance dans la prise de décision à l’instant t et peut-être pour l’instant t+1.
* La porte de sortie va fournir une décision à la fois pour le bloc LSTM suivant ainsi que pour l’utilisateur.

## Les deux modèles :

Dans le cadre de ce projet, j’ai réalisé deux modèles afin de comparer l’impact du pré-entrainement des mots lorsqu’on utilise du Word Embegging :

* Modèle 1 : LSTM avec Word Embedding non-pré-entrainé.
* Modèle 2 : LSTM avec Word Embedding pré-entrainé (GloVe).

Avec GloVe, on s’attend à de meilleurs résultats car les vecteurs de mots seront déjà pré-entrainée avec un grand nombre de données.

## Apprentissage du modèle neuronal

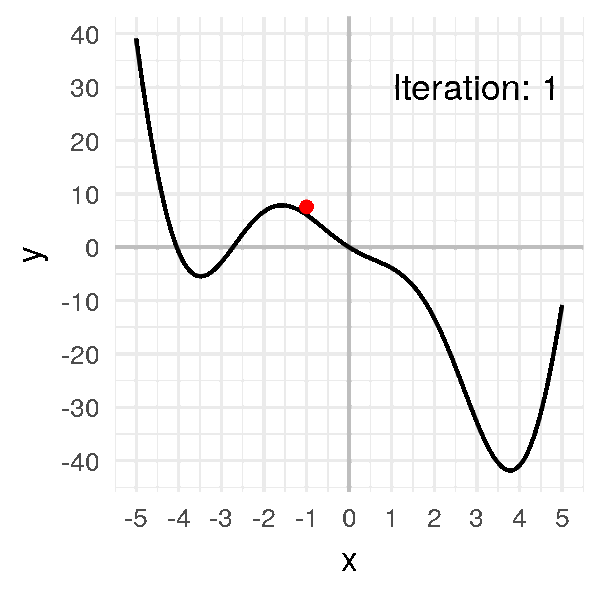
Pour l’apprentissage du modèle, j’ai utilisé la descente de gradient stochastique (SGD). Le pseudo-code est le suivant :

Une image contenant texte

Description générée automatiquement

Figure 2 : Pseudo-code d'une SGD

Le principe de la descente de gradient est de trouver le minimum d’une fonction à chaque itération on calcule la pente et on déplace le point en conséquence (cf. vidéo 1)



Vidéo 1 : Descente de gradient

# Evaluation du POS Tagger

## Description des corpus

Au total, le data set est divisé en 3 parties :

* Le corpus d’apprentissage (80%) : Il sert à l’apprentissage du modèle, c’est ici que l’on utilise la descente de gradient et qu’on l’optimise.
* Le corpus d’évaluation (10%) : Il sert à déterminer quand le modèle est le plus optimal, dans cette partie le gradient n’est pas utilisé.
* Le corpus de test (10%) : Il sert à tester le modèle le plus optimal sur un nouveau corpus.

## Description des métriques d’évaluations

Les différentes métriques d’évaluations sont la précision, le rappel et la F-mesure.

* Précision = TP / (TP+FP)
* Rappel : TP / (TP+FN)
* F-mesure = 2 x (Précision x Rappel) / (Précision + Rappel)

Avec :

* TP : Nombre de mots étiquetés correctement par le POS Tagger (les étiquettes de ces mots sont les mêmes que les étiquettes de la référence).
* FN : Nombre de mots qui n’ont pas été étiquetés par le POS Tagger (ces mots n’ont pas été étiquetés par le POS Tagger pourtant ils existent dans la référence).
* FP : Nombre de mots étiquetés pas correctement par le POS Tagger (les étiquettes de ces mots sont différentes des étiquettes de la référence).

Pour ce projet, j’ai seulement implémenter la précision, les autres n’ont pas pu l’être car faute de temps.

## Résultats

Les résultats ont été fait pour 10 époques, au-delà les valeurs n’évoluent plus significativement. (Cf. Annexe pour les résultats par époque)

|  |  |  |
| --- | --- | --- |
|  | Avec GloVe | Sans GloVe |
| Loss train set |  |  |
| Loss eval set |  |  |
| Précision Train set | 97,07% | 96,81% |
| Précision Eval set | 78,15% | 72,00% |
| Précision Test set | 81,91% | 70,48% |
| Temps d’epoque en Seconde | 8 | 8 |

On observe que l’utilisation de mot pré-entrainée améliore grandement les performances du modèle.

# Points forts

Les points forts de ce POS Tagger sont :

* L’utilisation de GloVe permet de grandement améliorer les performances du modèle.
* L’utilisation d’une architecture LSTM pour les prédictions est très efficace avec du Word Embedding.

# Limites du modèle

Les limites du modèle sont :

* La taille du dataset, dans le cadre d’une amélioration, il serait possible de prendre un data set plus grand ou alors de dupliquer celui existant.
* J’aurais pu aussi améliorer le modèle en utilisant une LSTM bidirectionnelle avec embedding comme ci-dessous :

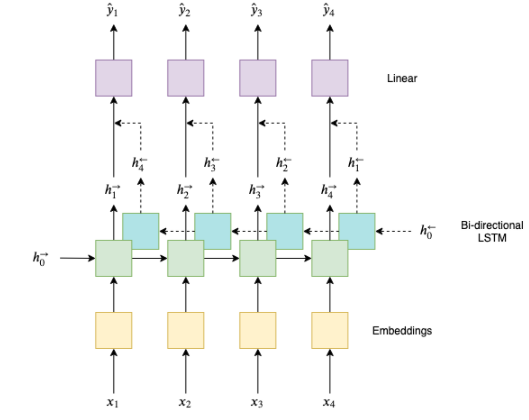


Figure 3 : Image d'un modèle avec une LSTM bidirectionelle

# Difficultés rencontrées

J’ai rencontré des difficultés concernant :

* La compréhension du fonctionnement d’une architecture LSTM.
* L’implémentation du Word Embedding avec Glove.
* Le temps accordé pour le projet.

# Organisation

Ce projet a été réalisé en solo, j’ai donc réalisé l’intégralité de ce dernier.

# Conclusion

L’utilisation des POS Tagger comme avec des plateformes comme Stanford Core NLP et NLTK est très répandue dans le domaine du NLP. Il est donc très intéressant d’en faire un pour comprendre le fonctionnement de cet outil.

Plus personnellement, j’ai adoré travailler sur ce projet. Cependant, j’aurais aimé pouvoir approfondir ce projet, étant seul et ayant une durée de projet assez courte, je n’ai pas pu aller au bout des choses.

# Annexe

Une image contenant texte

Description générée automatiquement

Figure 4 : Résultat du train + evaluate sur 10 époques

Une image contenant texte

Description générée automatiquement

Figure 5 : Résultat du train + evaluate sur 10 époques