Rapport PSTALN

Cléa Han, Yanis Labeyrie et Adrien Zabban

janvier 2024

Table des matières

1 Introduction

Le but de ce projet est de faire un modèle de langage capable de prédire les morphologies des mots d'une phrase (*morphy*), comme le montre la Figure 1.

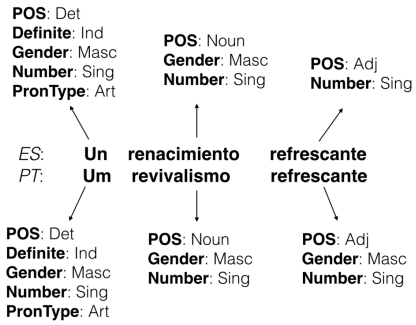

FIGURE 1 – Tag morphologique d'une phrase portugaise et sa traduction en espagnol. Image tirée de [1]

Nous allons aussi nous demander si est-ce que le fait d'ajouter un prétraitement de la phrase va améliorer les performances. L'idée de ce prétraitement est de faire prédire le balisage de séquence prototypique (*pos*) des mots, comme le montre la Figure 2.

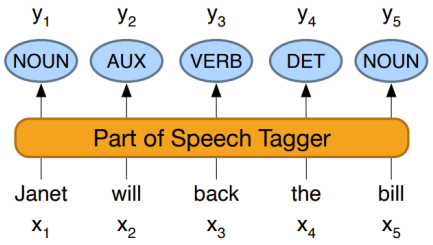

FIGURE 2 – La tâche d'étiquetage des parties du discours : la mise en correspondance des mots d'entrée $x_1, x_2, ..., x_n$ avec les étiquettes *pos* de sortie $y_1, y_2, ..., y_n$. Image tirée de [2]

2 Données

Nous avons utilisé le dataset Universal Dependencies 2.13 [3], qui comporte 146 langues dont l'anglais et le français. Dans ce projet, nous nous sommes seulement concentrés sur le français. Le dataset en français contient un ensemble de 47498 phrases, avec 849476 mots, et 76048 mots uniques. Ce dataset possède des phrases, la liste de mots composant ses phrases, et les *pos* et *morphy* qui sont associés à chacun de ses mots. Au total, nous avons recensé 19 *pos* et 28 *morphy* différents.

2.1 Padding

Les phrases données au modèle doivent être toutes exactement de la même longueur pendant l'entraînement. Notons K la longueur des séquences [1]. Nous créons donc des tokens <PAD> pour équilibrer les longueurs des phrases. Nous avons choisi d'utiliser une méthode naïve pour créer nos séquences qui consiste à couper chaque phrase pour former les séquences de K mots, et de rajouter si besoin des tokens <PAD> pour terminer la dernière séquence. Nous avons choisi de prendre $K = 10$.

2.2 Gestion des mots inconnus

Nous avons, avant l'entraînement du modèle, appris un vocabulaire de mots qui fait la correspondance entre les mots et un nombre unique. Le vocabulaire a été fait seulement sur les mots qui sont dans la base de données d'entraînement. C'est donc pour cela qu'il peut arriver que des mots de la base de données de validation ou de teste peuvent ne pas être reconnue par le vocabulaire et donc le modèle. Pour gérer ces mots inconnus, nous avons décidé de leur attribuer le token particulier : <UNK>. Pour que le modèle apprenne l'embedding de ce mot, il faut alors rajouter artificiellement des mots inconnus dans le corpus d'entraînement. Nous avons donc, pour chaque mot de ce corpus, remplacé par <UNK> avec une probabilité de 1%.

2.3 Encodage des labels

Pour encoder les *pos*, nous avons seulement créé une liste de tous les *pos* et avons encodé les *pos* avec leurs indices dans la liste. Nous avons aussi ajouté le *pos* <PAD> pour que le token de padding soit catégorisé dans ce *pos*.

Pour les *morphy*, cela a été plus compliqué, car un mot peut avoir plusieurs *morphy* associé, avec des valeurs différentes. Nous avons décidé d'encoder une suite de *morphy* par une liste de nombre de longueurs 28 et les éléments sont les indices des possibilités de chaque *morphy*. Par exemple : le label *Emph=No|Number=Sing|Person=1|PronType=Prs* est encodé par la liste ci-dessous :

[0, 0, 1, 0, 1, 2, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Comme pour le *pos*, nous avons aussi ajouté un *morphy* pour le padding. Étant donné que le *morphy* qui possède le plus de possibilités en possède 13, quand nous encodons les labels en one-hot, nous avons un tensor de shape (19) pour les *pos* et un tensor de shape $(28, 13)$ pour les *morphy*.

3 Modèle

3.1 Architecture des *pos*

Le modèle, que nous appelons *GET_POS*, est constitué d'une couche d'embedding pour apprendre les plongements des mots. Les données passent alors dans une couche LSTM bidirectionnel, puis dans une couche dense (fully connected layers), avec une sortie à 19 éléments représentant les probabilité de chaque classe *pos*. Nous avons utilisé du dropout sur les neurones des couches LSTM

1. Nous appelons séquence les suites de mots de même taille.

et dense, avec un taux d'oublie de 1%. Entre ces 3 couches, nous avons aussi ajouté la fonction d'activation ReLU [4]. Nous avons utilisé la CrossentropyLoss comme fonction de coût, l'optimizer Adam [5]. Ce modèle est représenté sur la Figure 3. Nous avons donc en entrée une matrice de taille $B \times K$, contenant l'indice des mots, où B est la taille du batch. Et le modèle retourne un tensor de taille $B \times K \times 19$, contenant les probabilités des *pos* pour chaque mot.

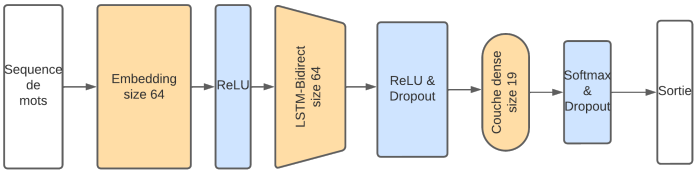

FIGURE 3 – Modèle *GET_POS*

3.2 Architecture des *morphy*

Pour la tâche de prédiction des traits morphologiques, la difficulté est supérieure. En effet, pour cette tâche il faut pour chaque mot de la phrase prédire à la fois son rôle dans la phrase (Verbe, Nom, etc.) mais aussi pour chacune de ses classes prédire des attributs (singulier, pluriel, etc.). Nous avons donc créé plusieurs architecture différente que nous allons présenter. Nous avons aussi dû modifier la fonction de coût par la moyenne des crossentropy sur les 28 *morphy* différents. Nous avons utilisé l'optimizer Adam [5], et avons ajouté un gradient decay, qui fait que le learning rate est divisé par deux lors des epochs 10 et 20.

3.2.1 Modèle *SUPERTAG*

Le modèle commence par une couche d'embedding pour deux couches de LSTM bidirectionnel. Ensuite, on fait passer les données dans 3 couches denses cachées dont la dernière contient 364 neurones [2] Avant que nos données sortent du modèle, elles ont une taille de $B \times B \times 364$. On *reshape* alors les données pour avoir une sortie de taille $B \times B \times 28 \times 13$. La Figure 4 représente ce modèle.

3.2.2 Modèle *SEPARATE*

On reprend le même début que le modèle *SUPERTAG* et l'on change la dernière couche. Au lieu de prédire tous les *morphy* en même temps avec une seule couche dense, on va avoir 28 couche dense (une par *morphy*). Chaque couche va alors prédire les probabilités liées à son *morphy*. On va alors ajouter, à chaque prédiction des couches, des $-\infty$ pour obtenir un vecteur de taille 13 pour tous les mots [3]. On va ensuite concaténer tous les résultats avec une sortie de taille $B \times B \times 28 \times 13$. La Figure 5 représente ce modèle.

2. Nous avons pris 364 neurones car $364 = 28 \times 13$.
3. On mets $-\infty$ car cela permet d'avoir une probabilité de 0 après le softmax.

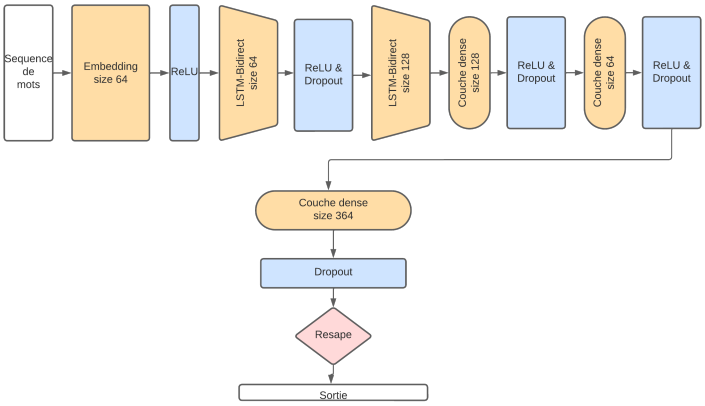

FIGURE 4 – Modèle *SUPERTAG*

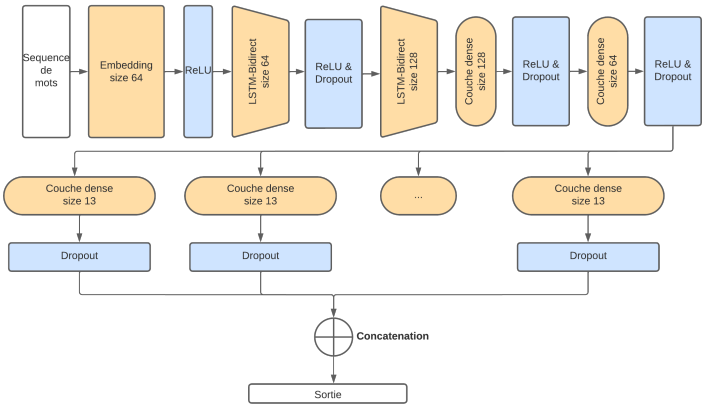

FIGURE 5 – Modèle *SEPARATE*.

3.2.3 Modèle *FUSION*

L'idée de ce modèle est d'utiliser la prédiction du modèle *GET_POS* pour aider le modèle *SEPARATE*. On fait passer les données dans les 2 couches de LSTM, et en sortie on va concaténer la sortie du LSTM avec la sortie d'un modèle *GET_POS* pré-entraînemé. La Figure 6 représente ce modèle.

4 Métriques

Pour pouvoir mesurer les performances de ces modèles, nous avons mis en place plusieurs métriques. Pour la prédiction de *pos*, nous avons utilisé l'accuracy micro et macro. La première nous donne l'accuracy de la prédiction et la deuxième nous donne la moyenne de l'accuracy sur chacune des classes.

Pour la prédiction de *morphy*, nous avons aussi utilisé l'accuracy micro, et nous avons aussi implémenté une métrique qu'on a appelée *all good*, qui fait la moyenne des mots dont la prédiction de tous les *morphy* sont juste. Si la

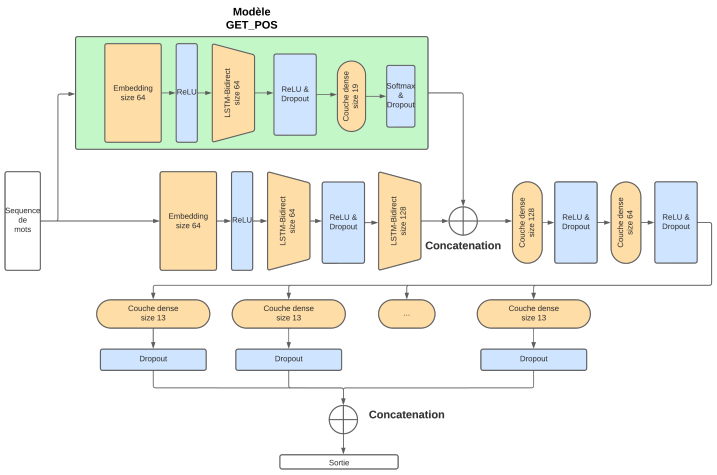

FIGURE 6 – Modèle *FUSION*

métrique vaut 0.2, cela veut dire qu'il y a 1 mot sur 5, qui dont la prédiction est totalement bonne.

5 Résultats

5.1 Résultats pour la prédiction de *pos*

Nous avons donc lancé un entraînement de notre réseau LSTM-Bidirectionel sur 30 epochs pour le *pos*-tagging et nous avons obtenu une accuracy de validation d'environ 90% ce qui dénote que le réseau a réussi à apprendre correctement cette tâche d'étiquetage, comme le montre la figure suivante :

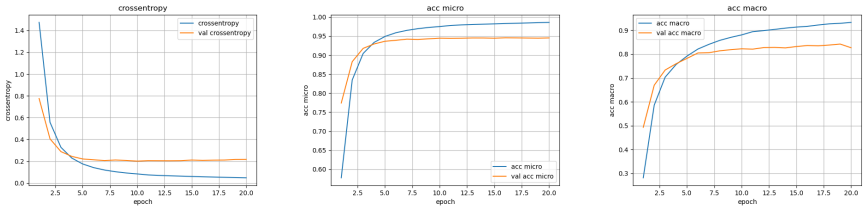

FIGURE 7 – Valeurs de la loss et des métriques d'entraînement et de validation en fonction des epochs.

Les courbes de loss et d'accuracy décrivent un apprentissage efficient dès la 5e epoch. À partir de la 5e epoch, l'apprentissage stagne et si on s'étend à plus de 20 epochs, on risque de faire du overfitting, car l'écart entre les performances de l'entraînement et les performances de la validation commence à se creuser. L'accuracy micro est meilleure que l'accuracy macro, ce qui signifie que si les performances de toutes les classes confondues sont meilleures que si on cherche à quantifier les performances globales à partir de chaque classe séparément, traitée de manière égale.

Nom du modèle	crossentropy	accuracy micro	accuracy macro
GET_POS	0.204	0.944	0.816

TABLE 1 – Résultats du modèle *GET_POS* sur la base de données de teste.

On constate que la valeur de l'accuracy de validation est un peu plus faible que celle d'entraînement, cela s'explique par plusieurs facteurs, notamment la difficulté du modèle à généraliser aux nouveaux mots inconnus.

5.2 Résultats la prédiction des *morphy*

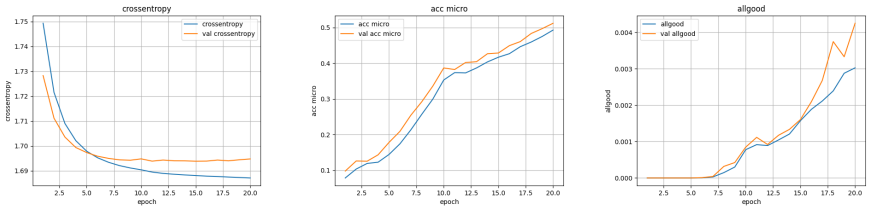

FIGURE 8 – Valeurs de la loss et des métriques d'entraînement et de validation en fonction des epochs pour le modèle *SUPERTAG*.

Nom du modèle	crossentropy	accuracy micro	all good
BASELINE	-	0.980	0.791
SUPERTAG	1.700	0.436	0.002
SEPARATE			
FUSION			

TABLE 2 – Résultats de test sur la prédiction des *morphy*

6 Inférences

7 Annexes

Mettre la liste des POS et MORPHY ici

Références

[1] C. Malaviya, M. R. Gormley et G. Neubig, "Neural Factor Graph Models for Cross-lingual Morphological Tagging," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, I. Gurevych et Y. Miyao, éd., Melbourne, Australia : Association for Computational Linguistics, juill. 2018, p. 2653-2663. doi : 10.18653/v1/P18-1247. adresse : https://aclanthology.org/P18-1247.

[2] D. Jurafsky et J. H. Martin, "Sequence Labeling for Parts of Speech and Named Entities," in *Speech and Language Processing*, P. Hall, éd., mai 2008. adresse : https://web.stanford.edu/~jurafsky/slp3/8.pdf.

[3] D. Zeman, J. Nivre, M. Abrams et al., *Universal Dependencies 2.13*, LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University, 2023. adresse : http://hdl.handle.net/11234/1-5287.

[4] A. F. Agarap, "Deep Learning using Rectified Linear Units (ReLU)," *CoRR*, t. abs/1803.08375, 2018. arXiv : 1803.08375. adresse : http://arxiv.org/abs/1803.08375.

[5] D. P. Kingma et J. Ba, "Adam : A method for stochastic optimization," *arXiv preprint arXiv :1412.6980*, 2014. adresse : https://arxiv.org/abs/1412.6980.