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.

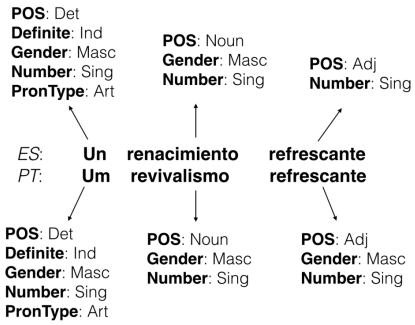


FIGURE 1 – Tag morphologique d'une phrase Portugaise et sa traduction en Espagnol. Image tirée de [1]

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.

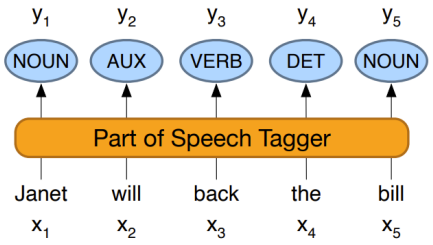


FIGURE 2 – La tâche d'étiquetage des parties du discours : la mise en correspondance des mots d'entrée $x_1, x_2, ..., x_n$ avec les étiquettes *pos* de sortie $y_1, y_2, ..., y_n$. Image tirée de [2]

2 Dataset d'entraînement et préparation des 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. **dire combien il y a de phrase et de mots en français**. 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.

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

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

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 longueur 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 données 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 Méthodologie et Modèle

3.1 Architecture du modèle

Pour réaliser cette tâche nous avons opté pour l'utilisation d'un réseau de neurone récurent de type LSTM (Long-Short Term Memory) bidirectionnel. En effet, cette approche classique est intéressante car le résau LSTM bidirectionnel permet d'analyser les dépendances entre les mots potentiellement éloignés d'une phrase, or ceci est nécessaire pour démêler les ambiguïtés lors de l'étiquetage morpho-syntaxie. Considérons l'exemple suivant :

Phrase : "Je ne peux pas ouvrir le fichier que tu m'as envoyé."

Sans la capacité de capturer les dépendances à long terme, une approche classique pourrait avoir du mal à démêler l'ambiguïté dans la phrase, en particulier en ce qui concerne le mot "que". Est-ce une conjonction de subordination introduisant une proposition subordonnée relative, ou est-ce une conjonction de coordination ?

Avec un réseau LSTM bidirectionnel, on peut mieux gérer ces dépendances à long terme. Le réseau peut prendre en compte le contexte des mots précédents et suivants pour déterminer que "que" est utilisé comme une conjonction de subordination dans ce cas précis. L'utilisation de la bidirectionnalité permet au modèle d'avoir une compréhension contextuelle plus riche, en analysant à la fois les mots antérieurs et postérieurs pour chaque position dans la séquence.

Le réseau LSTM que nous avons employé est constitué d'un module LSTM bidirectionnel (comprenant 2 couches avec 2 directions opposées) ainsi que d'une couche dense de $[n_{neurone}]$ qui récupère l'état caché de la dernière cellule LSTM est produit le vecteur des prédictions.

3.1.1 Architecture pour le *pos*-tagging

Nous utilisons la CrossentropyLoss pour comparer les vecteurs de densité de probabilité de classe prédit par le modèle (un pour chaque mot de la phrase avec le vecteur one-hot correspondant à la vrai classe du mot). Nous prédisons un nombre $[n_{classes}]$ d'attributs de *pos*-tagging possibles.

3.1.2 Architecture pour la prédiction de traits morphologiques

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, ect..) mais aussi pour chacune de ses classes prédire des attributs (singulier, pluriel, ...).

Pour cela une première méthode serait d'entraîner un modèle distinct par attributs, mais cela demanderai un assez long temps d'entraînement. Nous avons donc opté pour une première méthode différente : prédire non pas un vecteur comme pour la première tâche mais une matrice [classe,attribut] sans changer l'architecture du réseau, simplement en changeant le nombre de neurones de la couche dense. Nous avons par la suite tenté une deuxième approche : diviser la sortie de la couche dense en la faisant passer dans un nombre n de couche dense égal au nombre de classes, chacune chargée de prédire un attribut et reformer la matrice après passage dans ces couches distinctes. L'intéret de cette approche est que l'on utilise les features extraites du bloc LSTM de manière commune pour les sous-tâches mais chaque sous-tâche de classification est réalisé par une couche dense distincte.

La fonction de coût utilisé est toujours la CrossentropyLoss mais elle est appliquée cette fois-ci classe par classe puis ensuite moyennée pour obtenir la valeur de loss globale.

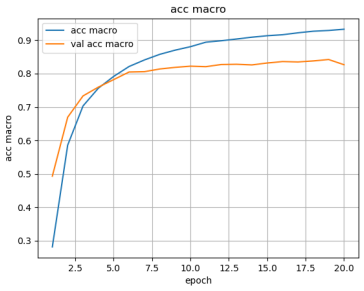
3.1.3 Problèmes rencontrés / Subtilités techniques

4 Résultats et métriques

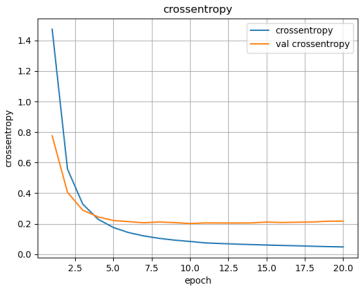
Métriques

4.1 Résultats et métriques pour le *pos*-tagging

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 à réussi à apprendre correctement cette tâche d'étiquetage, comme le montre la figure suivante :



(a) Valeurs de l'accuracy d'entraînement et de validation par epochs.



(b) Valeurs de la fonction de coût par epochs.

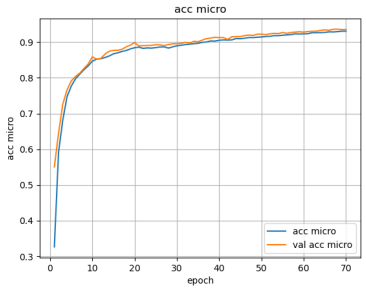
FIGURE 3 – Comparaison des récompenses cumulées sur 100 étapes

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.

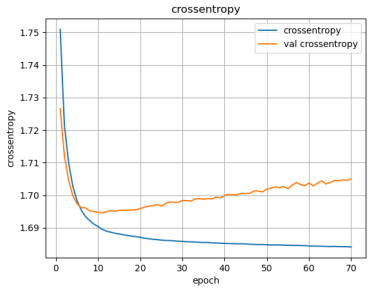
4.2 Résultats et métriques pour la prédiction des traits morphologiques

La tâche de prédiction des traits morphologiques comporte une subtilité : toutes les classes n'ont pas le même nombre d'attributs. Par exemple la classe "VERB" a 5 attributs tandis que la classe "NOUN" en a 12. Or, le réseau ne peut pas prédire un vecteur de taille variable. Pour résoudre ce problème, nous avons donc décidé de rajouter des attributs fictifs à certaines classes pour que toutes les classes aient le même nombre d'attributs. Pour ce faire deux solutions sont possibles : soit chacune des sous-couches dense du réseau a le même nombre de neurones, et dans ce cas on rajoute des attributs fictifs et inutiles à certaines classe, soit on définit des couches dense de tailles différentes pour chaque classe égales au noombre d'attributs de la classe, mais dans ce cas, il faut rajouter du padding à la sortie de la couche dense pour que toutes les sorties aient la même taille. Nous avons testé les deux méthodes (qu'on appelera "$separate$" pour la méthode sans padding et "$separate_0$" pour la méthode avec padding après couche et nous avons obtenu de meilleurs résultats avec la deuxième méthode :

Pour la tâche de prédiction des traits morphologiques avec la méthode $separate_0$, nous obtenons les résultats suivants :



(a) Valeurs de l'accuracy d'entraînement et de validation par epochs.



(b) Valeurs de la fonction de coût par epochs.

FIGURE 4 – Courbes d'accuracy et de crossentropy pour la tâche de prédiction des traits morphologiques

On mesure également une autre métrique appelée 'allgood' qui mesure le nombre de mots dont tous les attributs ont été prédits correctement. On obtient une valeur de allgood de : VALEUR comme le montre la figure suivante :

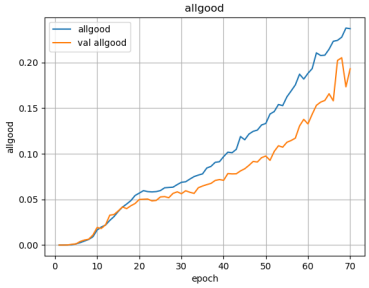
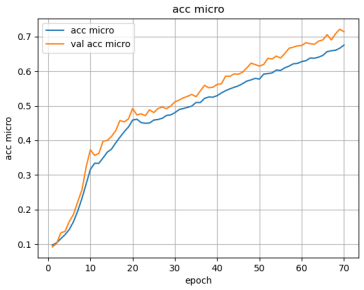


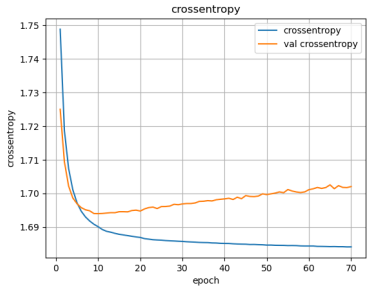
FIGURE 5 – Valeurs de la métrique allgood par epochs.

L'entraînement requiert un temps plus long que pour la tâche de *pos*-tagging car il y a beaucoups plus d'attributs à prédire. On obtient une accuracy de validation final de VALEUR, et les courbes montrent que le modèle subit un léger sur-apprentissage (comme le montre la valeur de la loss de validation).

Pour la tâche de prédiction des traits morphologiques avec la méthode *separate*, nous obtenons les résultats suivants :



(a) Valeurs de l'accuracy d'entraînement et de validation par epochs.



(b) Valeurs de la fonction de coût par epochs.

FIGURE 6 – Courbes d'accuracy et de crossentropy pour la tâche de prédiction des traits morphologiques

On constate que les résultats sont moins bons que pour la méthode $separate_0$ et que le modèle a besoin de plus d'epochs. Cela s'explique par le fait que les attributs fictifs rajoutés aux classes pour que toutes les classes aient le même nombre d'attributs sont des attributs inutiles qui n'apportent pas d'information au modèle et qui peuvent donc le perturber.

En revanche pour la tâche de prédiction des traits morphologiques les résultats se sont avérés bien inférieurs avec une accuracy de validation de seulement 20%.

Mettre ici le graphe d'entraînement

Mettre ici des exemples d'inférences (un cas simple, un cas difficile ambigu

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.