

# Rapport PSTALN

Cléa Han, Yanis Labeyrie et Adrien Zabban

janvier 2024

## Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Données</b>	<b>2</b>
2.1	Padding . . . . .	3
2.2	Gestion des mots inconnus . . . . .	3
2.3	Encodage des labels . . . . .	3
<b>3</b>	<b>Modèle</b>	<b>3</b>
3.1	Architecture des <i>pos</i> . . . . .	3
3.2	Architecture des <i>morphy</i> . . . . .	4
3.2.1	Modèle <i>SUPERTAG</i> . . . . .	4
3.2.2	Modèle <i>SEPARATE</i> . . . . .	5
3.2.3	Modèle <i>FUSION</i> . . . . .	5
<b>4</b>	<b>Mesurer les performances des modèles</b>	<b>6</b>
4.1	Mes métriques . . . . .	6
4.2	La baseline . . . . .	6
<b>5</b>	<b>Résultats</b>	<b>6</b>
5.1	Résultats pour la prédiction de <i>pos</i> . . . . .	6
5.2	Résultats la prédiction des <i>morphy</i> . . . . .	7
<b>6</b>	<b>Inférences</b>	<b>8</b>
<b>7</b>	<b>Conclusion</b>	<b>9</b>
<b>8</b>	<b>Annexes</b>	<b>10</b>

# 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. L'ensemble des *morphy* est recensé dans les Annexes (section 8, Figure 12).

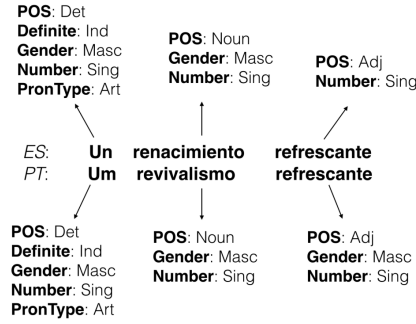


FIGURE 1 – Tag morphologique d'une phrase portugaise et sa traduction en espagnol. Image tirée de [1]

Nous allons aussi nous demander si 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. L'ensemble des *pos* est recensé dans les Annexes (section 8, Figure 11).

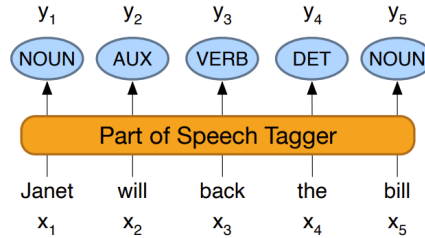


FIGURE 2 – La tâche d'étiquetage des parties du discours : la mise en correspondance des mots d'entrée  $x_1, x_2, \dots, x_n$  avec les étiquettes *pos* de sortie  $y_1, y_2, \dots, y_n$ . Image tirée de [2]

## 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, 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<sup>1</sup>. 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 test ne soient pas reconnus par le vocabulaire et donc par 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 remplacé chaque mot de ce corpus 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 *morph*, cela a été plus compliqué, car un mot peut avoir plusieurs *morph* associés, avec des valeurs différentes. Nous avons décidé d'encoder une suite de *morph* par une liste de nombre de longueurs 28 et les éléments sont les indices des possibilités de chaque *morph*. 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]
```

Comme pour le *pos*, nous avons aussi ajouté un *morph* pour le padding. Étant donné que le *morph* 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 *morph*.

# 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'oubli 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.

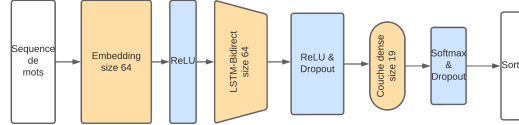


FIGURE 3 – Modèle *GET\_POS*

## 3.2 Architecture des *morphy*

Pour la tâche de prédiction des traits morphologiques, la difficulté est supérieure. En effet, pour cette tâche il faut prédire pour chaque mot de la phrase 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 architectures différentes 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<sup>2</sup>. 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.

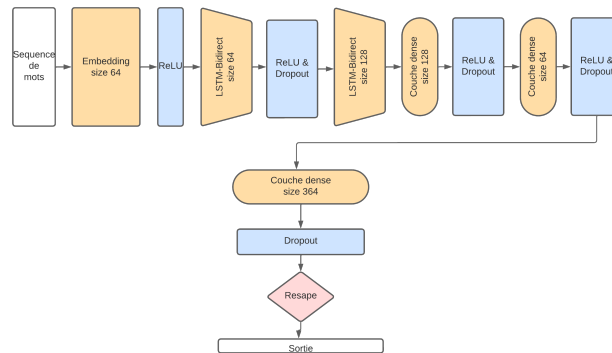


FIGURE 4 – Modèle *SUPERTAG*

2. Nous avons pris 364 neurones car  $364 = 28 \times 13$ .

### 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 couches denses (une par *morphy*). Chaque couche va alors prédire les probabilités liées à son *morphy*. On va alors ajouter, à chaque prédiction des couches, des  $-1000$  pour obtenir un vecteur de taille 13 pour tous les mots<sup>3</sup>. 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.

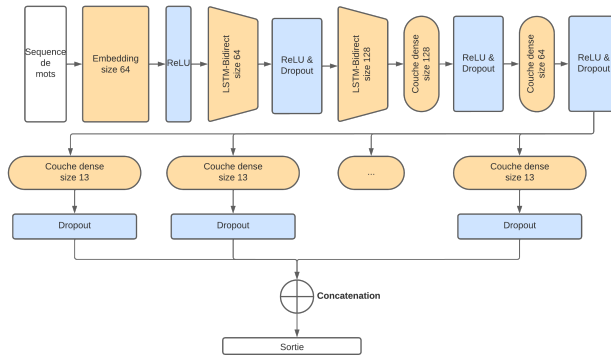


FIGURE 5 – Modèle *SEPARATE*.

### 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-entraîné. La Figure 6 représente ce modèle.

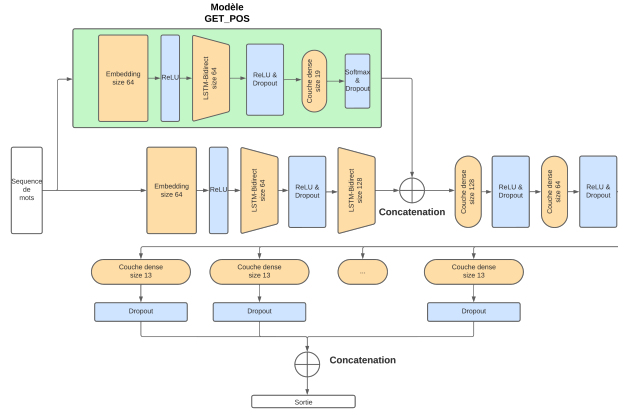


FIGURE 6 – Modèle *FUSION*

3. On a voulu mettre des  $-\infty$  car cela permettait d'avoir une probabilité de 0 après le softmax. Cependant, lorsque l'on faisait cela, on avait une loss avec une valeur de *not a number*, donc on a décidé de mettre  $-1000$  à la place.

## 4 Mesurer les performances des modèles

### 4.1 Mes 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 justes. Si la métrique vaut 0.2, cela veut dire qu'il y a 1 mot sur 5 dont la prédiction est totalement bonne.

### 4.2 La baseline

Nous avons implémenté une *BASLINE*, qui prédit les *morphy*, afin d'effectuer des comparaisons de performances avec nos modèles. Cette baseline s'appuie sur un modèle simple où un dictionnaire est créé et parcourt l'ensemble des données d'entraînements afin de récolter tous les mots qu'il n'a jamais croisés et leur label. Ainsi, le dictionnaire indique le vocabulaire du texte en lui assignant le premier label croisé lors du parcours du dataset d'entraînement.

## 5 Résultats

### 5.1 Résultats pour la prédiction de *pos*

Nous avons donc lancé un entraînement de notre réseau *GET\_POS* sur 20 epochs et nous avons obtenu une accuracy de validation d'environ 95% ce qui dénote que le réseau a réussi à apprendre correctement cette tâche d'étiquetage, comme le montre la Figure 7 :

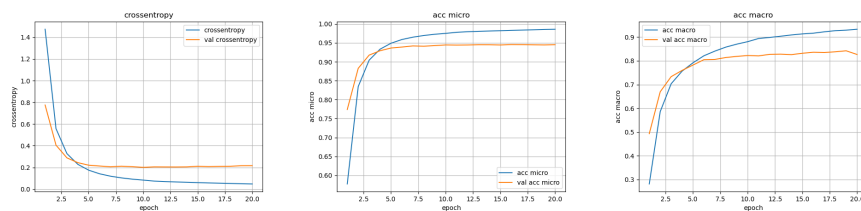


FIGURE 7 – Valeurs de la loss et des métriques d'entraînement et de validation en fonction des epochs.

Nous avons un apprentissage satisfaisant du modèle comme en témoignent les Figures 7. L'accuracy micro est relativement meilleure en termes de performances et de stabilité que l'accuracy macro.

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.

Nom du modèle	crossentropy	accuracy micro	accuracy macro
<i>GET_POS</i>	0.204	0.944	0.816

TABLE 1 – Résultats du modèle *GET\_POS* sur la base de données de teste.

Les résultats du Table 1 témoignent d’une performance satisfaisante du modèle POS avec une accuracy micro de 94,4%. Néanmoins, si on souhaite d’intéresser à l’accuracy micro, nous atteignons une accuracy de 81,6%.

## 5.2 Résultats la prédiction des *morphy*

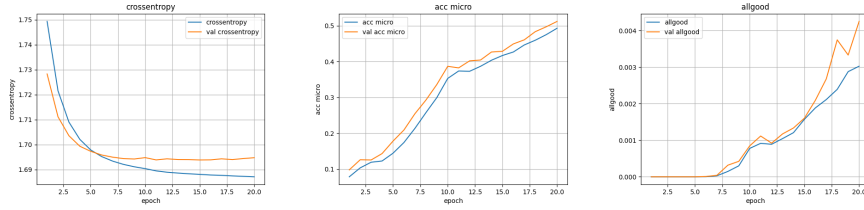


FIGURE 8 – Valeurs de la loss et des métriques d’entraînement et de validation en fonction des epochs pour le modèle *SUPERTAG*.

L’apprentissage est satisfaisant, car les courbes d’entraînement et de validation suivent la même allure d’après les Figures 4. On peut expliquer la meilleure performance de la validation par rapport à l’entraînement par l’utilisation du dropout dans l’apprentissage. Cette méthode est présente lors de l’entraînement, ce qui va naturellement baisser ses performances. Néanmoins, il y a un risque de overfitting lorsque l’on considère le modèle selon l’accuracy allgood.

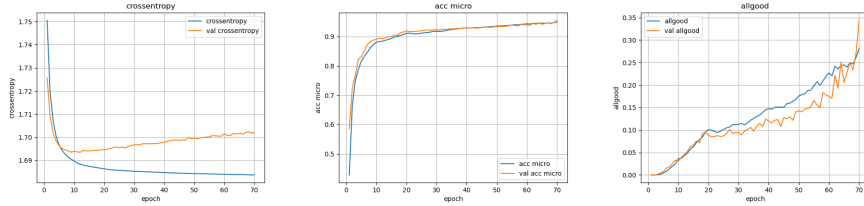


FIGURE 9 – Valeurs de la loss et des métriques d’entraînement et de validation en fonction des epochs pour le modèle *SEPARATE*.

Le modèle *SEPARATE* témoigne d’un apprentissage satisfaisant, où par rapport au modèle précédent, le risque d’overfitting semble plus faible selon l’accuracy et allgood, avec néanmoins de l’instabilité qui subsiste, d’après les Figures 9.

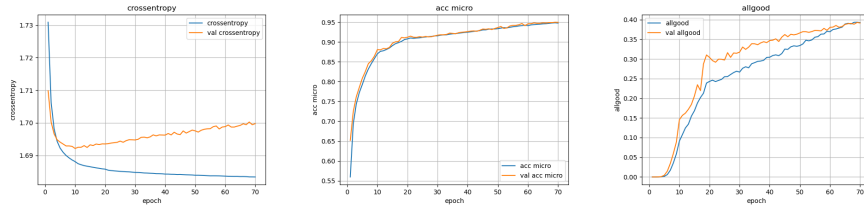


FIGURE 10 – Valeurs de la loss et des métriques d’entraînement et de validation en fonction des epochs pour le modèle *FUSION*.

Le modèle *FUSION* témoigne d’un apprentissage satisfaisant et relativement meilleur que celui des modèles précédents, à travers les Figures 6.

Nom du modèle	crossentropy	accuracy micro	all good
<i>BASLINE</i>	-	0.980	0.791
<i>SUPERTAG</i>	1.700	0.436	0.002
<i>SEPARATE</i>	1.70	0.893	0.046
<i>FUSION</i>	1.698	0.884	0.154

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

Les résultats du Table 2 mettent en comparaison les résultats de performances entre les modèles. Le modèle baseline indique une performance d’accuracy de 98% et donne les autres. Entre les trois autres modèles, c’est le modèle *SEPARATE* qui performe le mieux en termes d’accuracy micro, cependant, c’est le modèle *FUSION* qui performe le mieux en termes d’accuracy allgood.

Néanmoins, ces modèles ne permettent pas de dépasser les performances de la baseline, alors que les modèles implémentés ont une significative complexité plus importante que celle de la baseline. Ce qui se peut s’expliquer principalement parce que les *morph* des mots ne changent presque jamais, cela avantage énormément la baseline.

## 6 Inférences

Nous avons ensuite testé nos différents modèles en leurs faisant prédire les *morph* sur une phrase. Nous avons fait l’inférence sur la phrase : *Les bananes sont jaunes et mûrs.* Comme cette phrase n’a que 7 mots (6 mots et un point), on rajoute 3 caractères de padding <PAD>. Les Tables 3 et 4 représentent respectivement les *morph* prédit par les modèles du mot *bananes* et du dernier <PAD>.



modèle	inférences
<i>BASELINE</i>	Gender=Fem Number=Plur
<i>SUPERTAG</i>	NumForm=Roman Abbr=Yes Morph=VInf
<i>SEPARATE</i>	Gender=Neut Number=Plur VerbForm=Ger Degree=Pos Abbr=Yes
<i>FUSION</i>	Gender=Fem,Masc Number=Plur PronType=Tot NumType=Mult Case=Nom NumForm=Combi

TABLE 3 – Inférences du mot *bananes*

modèle	inférences
<i>BASELINE</i>	<PAD>=Yes
<i>SUPERTAG</i>	PronType=Emp Style=Coll
<i>SEPARATE</i>	<PAD>=Yes PronType=Int,Rel Case=Nom Degree=Sup NumForm=Combi
<i>FUSION</i>	<PAD>=Yes Gender=Neut PronType=Int,Rel NumType=Mult Degree=Sup Style=SIng NumForm=Combi

TABLE 4 – Inférences du dernier <PAD>.

Le modèle *SEPARATE*, pour le dernier <PAD>, prédit bien du <PAD> mais également d’autres labels qui n’ont rien à voir et qui sont incorrects. Une raison peut-être que la séparation des couches présentes dans notre modèle implique que certaines couches denses ne communiquent plus entre elles à une certaine étape de notre modèle pour réaliser leur inférence.

D’autre part, le modèle *FUSION* prédit un nombre superflu de labels de manière générale, plus que nécessaire. Le modèle combine non seulement cette architecture de séparation de couches denses pour différentes classes, mais également une fusion qui multiplie alors le nombre de labels inféré.

Dans cette perspective d’inférence, le modèle *SUPERTAG* semble relativement plus pertinent en fournissant un nombre relativement plus raisonnable, cependant, il reste non adapté étant donné que ces prédictions sont tout simplement incorrectes.

L’inférence fait ressortir différentes problématiques des architectures adoptées pour nos modèles.

## 7 Conclusion

Nous avons exploré deux tâches principales : l’étiquetage des parties du discours (*pos*) et la prédiction des traits morphologiques (*morph*).

Le modèle *GET\_POS* a démontré des performances impressionnantes dans la prédiction des parties du discours, avec une accuracy micro de 94.4% et une accuracy macro de 81.6% sur le jeu de test. Ces résultats témoignent de la capacité du modèle à apprendre efficacement la structure grammaticale des phrases en français.

Pour la tâche de prédiction des traits morphologiques, trois architectures de modèles ont été développées : *SUPERTAG*, *SEPARATE*, et *FUSION*. Chacun

de ces modèles a présenté des performances variées. Le modèle *SEPARATE* a obtenu une accuracy micro de 89.3% et une métrique *all good* de 4.6%, montrant sa capacité à prédire les traits morphologiques pour chaque classe. Cependant, le modèle *FUSION* a montré une instabilité lors de l'entraînement, ce qui peut nécessiter des ajustements pour améliorer ses performances.

Néanmoins, ces performances sont à prendre avec du recul et à nuancer, car nos résultats d'inférences sont assez décevants. Cela est sûrement dû à des problématiques intrinsèques aux architectures adoptées dans nos modèles, ce qui mérite une investigation plus poussée et des perspectives à étudier.

Les résultats obtenus ouvrent des perspectives intéressantes pour l'amélioration des modèles de langage, en particulier dans le contexte du Français. Des ajustements futurs pourraient inclure des stratégies plus sophistiquées pour la gestion des mots inconnus et des expériences avec d'autres architectures de modèles.

Ce projet a donc permis d'explorer différentes facettes de la modélisation du langage, de la prédiction des parties du discours à la prédiction des traits morphologiques. Les résultats obtenus constituent une base solide pour des développements futurs dans le domaine de la compréhension automatique du langage naturel en français.

## 8 Annexes

	Tag	Description	Example
Open Class	<b>ADJ</b>	Adjective: noun modifiers describing properties	<i>red, young, awesome</i>
	<b>ADV</b>	Adverb: verb modifiers of time, place, manner	<i>very, slowly, home, yesterday</i>
	<b>NOUN</b>	words for persons, places, things, etc.	<i>algorithm, cat, mango, beauty</i>
	<b>VERB</b>	words for actions and processes	<i>draw, provide, go</i>
	<b>PROPN</b>	Proper noun: name of a person, organization, place, etc..	<i>Regina, IBM, Colorado</i>
	<b>INTJ</b>	Interjection: exclamation, greeting, yes/no response, etc.	<i>oh, um, yes, hello</i>
Closed Class Words	<b>ADP</b>	Adposition (Preposition/Postposition): marks a noun's spacial, temporal, or other relation	<i>in, on, by, under</i>
	<b>AUX</b>	Auxiliary: helping verb marking tense, aspect, mood, etc.,	<i>can, may, should, are</i>
	<b>CCONJ</b>	Coordinating Conjunction: joins two phrases/clauses	<i>and, or, but</i>
	<b>DET</b>	Determiner: marks noun phrase properties	<i>a, an, the, this</i>
	<b>NUM</b>	Numeral	<i>one, two, first, second</i>
	<b>PART</b>	Particle: a function word that must be associated with another word	<i>'s, not, (infinitive) to</i>
	<b>PRON</b>	Pronoun: a shorthand for referring to an entity or event	<i>she, who, I, others</i>
Other	<b>SCONJ</b>	Subordinating Conjunction: joins a main clause with a subordinate clause such as a sentential complement	<i>that, which</i>
	<b>PUNCT</b>	Punctuation	<i>; , ()</i>
	<b>SYM</b>	Symbols like \$ or emoji	<i>\$, %</i>
	<b>X</b>	Other	<i>asdf, qwfg</i>

FIGURE 11 – Liste des labels *pos*. Image tirée de [2]

```

"_: ["Not", "Yes"],
"<PAD>": ["Not", "Yes"],
"Emph": ["Not", "No", "Yes"],
"Gender": ["Not", "Fem", "Masc", "Neut", "Fem,Masc"],
"Number": ["Not", "Sing", "Plur"],
"Person": ["Not", "3", "1", "2"],
"PronType": ["Not", "Prs", "Dem", "Ind", "Int", "Rel", "Art", "Emp", "Tot", "Neg"],
"Typo": ["Not", "Yes"],
"ExtPos": ["Not", "VERB", "PROPN", "NOUN", "ADV", "ADP", "ADJ", "SCONJ", "DET"],
"Tense": ["Not", "Pres", "Imp", "Past", "Fut"],
"VerbForm": ["Not", "Ger", "Fin", "Part", "Inf"],
"NumType": ["Not", "Ord", "Card", "Mult", "Frac"],
"Mood": ["Not", "Ind", "Imp", "Sub", "Cnd"],
"Poss": ["Not", "Yes"],
"Case": ["Not", "Acc", "Nom", "Gen"],
"Reflex": ["Not", "Yes"],
"Polarity": ["Not", "Neg", "Pos"],
"Degree": ["Not", "Cmp", "Pos", "Sup"],
"Style": ["Not", "Vrnc", "Expr", "Arch", "Slng", "Coll"],
"Number[psor]": ["Not", "Plur", "Sing"],
"Person[psor]": ["Not", "1", "3", "2"],
"NumForm": ["Not", "Roman", "Word", "Combi", "Digit"],
"Abbr": ["Not", "Yes"],
"Voice": ["Not", "Pass", "Act"],
"AdpType": ["Not", "Prep"],
"Foreign": ["Not", "Yes"],
"Definite": ["Not", "Ind", "Def"],
"Morph": ["Not", "VFin", "VInf", "VPar"]

```

FIGURE 12 – Liste des *morphy*

## 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](https://aclanthology.org/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](https://arxiv.org/abs/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>.