Actes de la 28e Conférence sur le Traitement Automatique des Langues Naturelles
Lille, France, 28 juin au 2 juillet 2021

Analyse en dépendances du français avec des plongements contextualisés

Loïc Grobol[1, 2, 3] Benoît Crabbé[1]
(1) LLF, CNRS, Université de Paris, 8, Rue Albert Einstein 75013 Paris, France
(2) Lattice, CNRS, ENS, PSL, Université Sorbonne Nouvelle, 1 Rue Maurice Arnoux, 92120 Montrouge, France
(3) LIFO, ICVL, Université d'Orléans, 45000 Orléans, France
loic.grobol@ens.psl.eu, benoit.crabbe@linguist.univ-paris-diderot.fr

RÉSUMÉ
Cet article présente un analyseur syntaxique en dépendances pour le français qui se compare favorablement à l'état de l'art sur la plupart des corpus de référence. L'analyseur s'appuie sur de riches représentations lexicales issues notamment de BERT et de FastText. On remarque que les représentations lexicales produites par FlauBERT ont un caractère auto-suffisant pour réaliser la tâche d'analyse syntaxique de manière optimale.

ABSTRACT
French dependency parsing with contextualized embeddings

This paper presents a dependency parser for French that compares favorably to the state of the art on several corpora. The parser relies on rich lexical representations from BERT and FastText. We notice that the lexical representations produced by FlauBERT are somehow self-sufficient to perform the syntax analysis task in an optimal way.

MOTS-CLÉS : Analyse syntaxique en dépendances du français, BERT, FastText.

KEYWORDS : Dependency Parsing, Parsing French, BERT, FastText.

1 Introduction

Cet article décrit un modèle d'analyse syntaxique en dépendances couplé à un étiqueteur morphosyntaxique pour le français [1]. Nous étudions plus spécifiquement l'impact de représentations lexicales apprises de manière non supervisée sur de gros volumes de texte telles que FastText et BERT en complément de plongements lexicaux standards. Nous commençons par expliciter le modèle d'analyse en section 2 avant de présenter les expériences sur les représentations lexicales en section 3.

2 Modèle d'analyse

Le modèle d'analyse combine à la fois un étiqueteur morphosyntaxique, un analyseur basé sur l'algorithme de Dozat & Manning (2017) et l'utilisation de riches représentations lexicales (Bojanowski *et al.*, 2017 ; Devlin *et al.*, 2019).

1. Disponible à https://github.com/bencrabbe/npdependency

Dans ce qui suit, on suppose qu'une phrase $w_1 ... w_n$ est représentée par une séquence de plongements lexicaux $\boldsymbol{X} = \boldsymbol{x}_1 ... \boldsymbol{x}_n$. En préalable à l'analyse nous traitons les plongements lexicaux à l'aide d'un LSTM pour construire une séquence de représentations contextualiséss $\boldsymbol{C} = \boldsymbol{c}_1 ... \boldsymbol{c}_n$:

$$\boldsymbol{C} = \text{LSTM}(\boldsymbol{X})$$

L'étiquetage morphosyntaxique est réalisé à l'aide d'un réseau à propagation avant (MLP) et d'une sortie SOFTMAX :

$$\hat{\boldsymbol{e}}_i = \textsc{softmax}(\text{MLP}^{\text{POS}}(\boldsymbol{c}_i))$$

$\hat{\boldsymbol{e}}_i$ est alors la distribution de scores des étiquettes morphosyntaxiques pour le mot w_i.

Pour l'analyse syntaxique, nous commençons par spécialiser les représentations de $\boldsymbol{C}$ de quatre manières différentes :

$$\begin{aligned}
\boldsymbol{h}_i^{\text{arc}-\text{dep}} &= \textsc{mlp}^{\text{arc}-\text{dep}}(\boldsymbol{c}_i) \\
\boldsymbol{h}_i^{\text{arc}-\text{gov}} &= \textsc{mlp}^{\text{arc}-\text{gov}}(\boldsymbol{c}_i) \\
\boldsymbol{h}_i^{\text{label}-\text{dep}} &= \textsc{mlp}^{\text{label}-\text{dep}}(\boldsymbol{c}_i) \\
\boldsymbol{h}_i^{\text{label}-\text{gov}} &= \textsc{mlp}^{\text{label}-\text{gov}}(\boldsymbol{c}_i)
\end{aligned}$$

où $\boldsymbol{h}_i^{\text{arc}-\text{gov}}$ (resp. $\boldsymbol{h}_i^{\text{arc}-\text{dep}}$) représente une spécialisation de la représentations de w_i du mot comme gouverneur (resp. comme dépendant). Ces représentations étant utilisées pour prédire les arcs, l'utilisation de représentations différentes pour les rôles de gouverneur et de dépendant permet d'éviter des effets de symétrie : le score d'un arc entre deux mots ne devrait en effet pas être le même selon que l'on choisisse soir l'un soit l'autre comme gouverneur. µ On réalise également une spécialisation analogue pour créer des représentations spécifiques pour la tâche d'étiquetage des arcs que l'on note $\boldsymbol{h}_i^{\text{label}-\text{dep}}$ et $\boldsymbol{h}_i^{\text{label}-\text{gov}}$.

La prédiction des arcs et de leurs étiquettes repose sur plusieurs fonctions biaffines de la forme :

$$\textsc{Biaff}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^T \boldsymbol{W} \boldsymbol{y} + \boldsymbol{U}(\boldsymbol{x} \oplus \boldsymbol{y}) + \boldsymbol{b}$$

On utilise une première fonction biaffine pour attribuer un score $s_{j \to i}^{arc}$ à chaque arc entre un gouverneur w_j et un dépendant w_i :

$$s_{j \to i}^{arc} = \textsc{Biaff}(\boldsymbol{h}_i^{\text{arc}-\text{gov}}, \boldsymbol{h}_j^{\text{arc}-\text{dep}})$$

et autant de fonctions biaffines qu'il y a de types de dépendances ℓ pour attribuer un score à chaque étiquette possible pour ce même arc :

$$s_{j \to i}^{\ell} = \textsc{Biaff}_{\ell}(\boldsymbol{h}_i^{\text{label}-\text{gov}}, \boldsymbol{h}_j^{\text{label}-\text{dep}})$$

Ces fonctions diffèrent les unes des autres, les matrices de paramètres $\boldsymbol{W}_{\ell}$ et $\boldsymbol{U}_{\ell}$ étant *a priori* distinctes pour chaque ℓ.

Fonction objectif La fonction objectif du modèle est composée d'objectifs multiples. Le premier porte sur l'étiquetage morphosyntaxique. On suppose qu'une phrase est un couple (W, E) composé d'une séquence de mots $W = w_1 ... w_n$ et d'une séquence d'étiquettes de référence $E = e_1 ... e_n$. Cette première fonction calcule l'entropie croisée entre la distribution prédite par le modèle $\hat{\boldsymbol{e}}_i$ pour chaque tag et la distribution ponctuelle $\boldsymbol{e}_i = (\delta_{ij})_j$ qui encode e_i :

$$\mathcal{L}(W, T) = \sum_{i=1}^{n} H(\boldsymbol{e}_i, \hat{\boldsymbol{e}}_i)$$

Le second objectif porte sur la prédiction des arcs, indépendamment de leurs étiquettes. Dans ce cas on suppose que la phrase est un couple (W,D), où D est un ensemble d'arcs de dépendance de référence. Pour chaque arc $(j \rightarrow i) \in D$ on construit un vecteur $\boldsymbol{g}_i = (\delta jk)_k$ qui indique la position du gouverneur w_j du dépendant w_i. On obtient du modèle une matrice $\hat{\boldsymbol{G}} = \hat{\boldsymbol{g}}_{i,j}$ telle que pour tout i $(\hat{\boldsymbol{g}}_{i,j})_j = \textsc{softmax}(s^{arc}_{1\rightarrow i} ... s^{arc}_{n\rightarrow i})$ et la fonction objectif est alors de la forme suivante :

$$\mathcal{L}(W,D) = \sum_{i=1}^{n} H(\boldsymbol{g}_i, \hat{\boldsymbol{g}}_i)$$

Le dernier objectif concerne l'étiquetage des dépendances. Celui-ci suppose un étiquetage de l'ensemble des arcs de référence par une fonction L. On code l'étiquetage de référence d'un arc $(j \rightarrow i) \in D$ par une étiquette $\ell = L(j \rightarrow i)$ par le vecteur $\boldsymbol{l}_{j\rightarrow i} = (\delta_{k\ell})_k$. Pour chaque arc $(j \rightarrow i) \in D$ on obtient du modèle un vecteur de score $\hat{\boldsymbol{l}}_{j\rightarrow i} = \textsc{softmax}(s^{\ell_1}_{j\rightarrow i} ... s^{\ell_k}_{j\rightarrow i})$. Pour une phrase donnée on calcule l'entropie croisée, c'est-à-dire :

$$\mathcal{L}(W,D,L) = \sum_{(j\rightarrow i)\in D} H(\boldsymbol{l}_{j\rightarrow i}, \hat{\boldsymbol{l}}_{j\rightarrow i})$$

Considérons un corpus arboré $T = ((W_i, E_i, D_i, L_i))_{1 \leq i \leq N}$ annoté en dépendances et étiqueté morphosyntaxiquement. La fonction objectif globale est la somme des trois objectifs définis ci-dessus :

$$\mathcal{L}(T) = \sum_{i=1}^{N} \mathcal{L}(W_i, E_i) + \mathcal{L}(W_i, D_i) + \mathcal{L}(W_i, D_i, L_i)$$

Le problème d'optimisation consiste alors à minimiser $\mathcal{L}(T)$. En pratique, nous utilisons la variante de descente de gradient stochastique appelée Adam (Kingma & Ba, 2014) et retenons le modèle qui minimise la perte sur le jeu de validation au cours d'un nombre e d'epochs fixé à l'avance.

Prédiction Pour prédire un arbre de dépendance à partir d'une phrase $w_1 ... w_n$ donnée en entrée, on commence par évaluer $s^{arc}_{j\rightarrow i}$ pour $1 \leq i \leq n$ et $1 \leq j \leq n$. La matrice de scores $\hat{\boldsymbol{G}}$ est vue comme la matrice de poids d'un graphe pondéré complet pour lequel on calcule une arborescence A couvrante de poids maximal à l'aide de l'algorithme de Chu-Liu/Edmonds (Chu & Liu, 1965 ; Edmonds, 1967). On assigne alors à chaque arc $(j \rightarrow i) \in A$ l'étiquette de score maximal :

2.1 Représentations lexicales

L'analyseur utilise des représentations lexicales de natures diverses, et la représentation $\boldsymbol{x}_i$ d'un mot w_i dans la phrase est la concaténation de représentations calculés par différents modèles lexicaux :

$$\boldsymbol{x}_i = \textsc{FastText}(w_i) \oplus \textsc{BERT}(w_i) \oplus \textsc{char-rnn}(w_i) \oplus \textsc{lookup}(w_i) \quad (1)$$

La représentation LOOKUP est une représentation vectorielle ne dépendant que de la forme lexicale de w_i et stockée dans un dictionnaire. La représentation au niveau des caractères (CHAR-RNN) est obtenu par encodage de la séquence des caractères du mot à l'aide d'un bi-LSTM (Hochreiter & Schmidhuber, 1997). Les paramètres de ces deux représentations sont initialisées aléatoirement et appris sur les données du treebank d'entraînement en même temps que les paramètres de l'analyseur.

En revanche, les représentations FASTTEXT (Bojanowski *et al.*, 2017) et BERT (Devlin *et al.*, 2019) sont obtenus à partir de modèles entraînés sur de beaucoup plus gros volumes de données, et

sont seulement ajustés sur le treebank d'entrainement. L'embedding FastText est la moyenne des embeddings $\boldsymbol{x}_s$ de l'ensemble $S(w_i)$ des sous mots qui composent le mot w_i. Les embeddings BERT sont calculés à l'aide d'une succession de réseaux Transformer (Vaswani *et al.*, 2017). Précisément, on utilise une séquence de transformations de la forme :

$$\begin{aligned} \boldsymbol{c}_i^0 &= \textsc{lookup}(w'_i) \oplus \textsc{position}_i \\ \boldsymbol{c}_i^{l+1} &= \textsc{Transformer}(\boldsymbol{c}_i^l; \boldsymbol{c}^l) \qquad (0 \leq l < 12) \\ \text{BERT}(w'_i) &= \frac{1}{12} \sum_l \boldsymbol{c}_i^l \end{aligned}$$

où position est une famille de vecteurs encodant des positions et où $\boldsymbol{w}'$ n'est plus une séquence de mots, mais une séquence de *sous*-mots fournie par un segmenteur appris automatiquement Enfin, comme représentation $\text{BERT}(w_i)$ du mot w_i, on choisit la moyenne des embeddings de ses sous-mots.

3 Expériences

Nous avons testé l'analyseur sur différents jeux de données, principalement issues du projet Universal Dependencies (Zeman *et al.*, 2020). Les expériences de développement sont réalisées sur le sous-corpus de développement du corpus UD_French-GSD (Guillaume *et al.*, 2019), deuxième plus grand corpus de français disponible dans UD. Celles-ci permettent à mettre en perspective, par ablation, l'impact des différents modèles de représentation lexicales. Elles nous ont également permis d'optimiser les hyperparamètres de nos modèles dans ces différentes configurations et de la procédure d'apprentissage. Ces optimisations ont été faites empiriquement, le nombre et le coût de ces expériences rendant une recherche systématique peu envisageable. Les résultats données pour ces expériences le sont sur le jeu de données de développement et non sur celui de test, afin d'éviter des effets de surapprentissage d'architecture.

Finalement nous testons sur les autres principaux corpus UD pour le français (UD_French-Sequoia (Candito & Seddah, 2012 ; Bonfante *et al.*, 2018) et UD_French-Spoken (Lacheret *et al.*, 2014 ; Gerdes & Kahane, 2017)) à l'exception du French Treebank, pour lequel nous reprenons la version utilisée pour la campagne d'évaluation SPMRL 2013 (Seddah *et al.*, 2013), ceci afin de pouvoir nous comparer plus facilement aux travaux existants. Tous les corpus UD sont utilisés dans leur version 2.7.

Nous avons utilisé principalement le modèle FlauBERT (Le *et al.*, 2020) comme implémentation du modèle BERT. Pour la tâche d'analyse syntaxique nous avons remarqué lors d'expériences préliminaires qu'il se comporte généralement un peu mieux que le modèle CamemBERT (Martin *et al.*, 2020) même si les différences sont dans la marge d'erreur. Les modèles BERT utilisés sont les versions `base-cased` de FlauBERT et de mBERT tels que distribués dans la version 4.2.2 de la bibliothèque Transformers (Wolf *et al.*, 2020).

3.1 Expériences de développement

Le modèle d'analyse syntaxique décrit ici se caractérise essentiellement par un enrichissement des représentations lexicales acquises à partir de gros volumes de données. L'ajout de ces représentations a également un coût non négligeable sur la taille du modèle et sur les temps d'exécution. Nous

TABLE 1 – Résultats (dev) des expériences d'ablations sur UD_FRENCH-GSD 2.7

FASTTEXT	FLAUBERT	Lookup	Caractères	UPOS	UAS	LAS	CLAS
+	+	+	+	98,61	96,74	95,51	92,84
-	+	-	-	98,56	96,73	95,56	92,87
+	-	+	+	97,72	93,67	91,65	87,02
+	-	+	-	97,81	93,70	91,64	86,96
+	-	-	+	97,36	92,66	90,34	85,18
+	-	-	-	97,31	92,94	90,58	85,61
-	-	+	+	96,72	92,61	90,27	84,81
-	-	+	-	96,74	92,75	90,26	84,86
-	-	-	+	95,54	91,26	88,32	82,23

Lookup et caractères sont neutralisés en leur affectant des plongements de taille 2 (ce qui les rend moralement inexploitables par le modèle sans nécessiter un changement d'architecture), les plongements FLAUBERT sont neutralisés en les supprimant complètement des entrées. Enfin, pour les plongements FASTTEXT, − signifie que les plongements utilisés n'ont pas été préentraînés.
Les scores rapportés ici sont obtenus en répétant la procédure d'entraînement avec cinq germes aléatoires et en conservant les résultats du modèle le plus performant (en terme de LAS) sur le corpus de développement. Ces scores sont calculés par le script d'évaluation officiel de la campagne d'évaluation CoNLL 2018 [2] dont nous conservons [3] les métriques UPOS, UAS, LAS, et CLAS (Nivre & Fang, 2017).

reportons ici quelques chiffres sur les performances de notre architecture en suivant une méthode d'ablation sur le corpus de développement. Les deux premières lignes du tableau 1 apportent un élément de réponse à la question : peut on se contenter de BERT comme seule représentation lexicale pour le parsing ? Il semble que la réponse est affirmative vu que la différence avec un modèle où toutes les autres représentations sont neutralisées est négligeable.

Notre seconde question est : peut-on se passer de BERT et obtenir des résultats équivalents en analyse syntaxique ? La troisième ligne du tableau donne notre meilleur modèle qui n'utilise pas de représentations de type BERT et contribue à apporter une réponse négative à la question.

En revanche, ce modèle utilise FASTTEXT et on remarque à partir des lignes 3 à 6 de la table que l'ablation additionnelle des représentations LOOKUP semble avoir un impact nettement plus significatif que l'omission des embeddings de caractères. Remarquons finalement à partir des lignes 7, 8 et 9 qu'utiliser des représentations FASTTEXT non-préapprises entraîne une dégradation significative des performances. La suppression du sous modèle de caractères a là encore un effet négligeable.

Dans l'ensemble on remarque que l'apport de BERT est décisif : près de 5 points en LAS par rapport à un modèle appris uniquement à l'aide de représentations de mots sur le corpus d'entraînement. L'apport de FastText est plus modeste, et le sous-modèle de caractère semble avoir une contribution nulle dans les différentes configurations testées.

Ces tendances sont par ailleurs cohérentes pour les métriques UAS, LAS, mais aussi et surtout pour CLAS, ce qui suggère qu'elles sont bien liées à de réelles différences de performances pour la reconnaissance des structures syntaxiques à l'échelle de la phrase plutôt qu'à des différence dans

2. https://github.com/ufal/conll2018/tree/e865d82e4c296d1660c9e7efcceb79aa418b6348
3. Les métriques MLAS et BLEX, bien que plus récentes et préférées à CLAS dans cette campagne d'évaluation, n'auraient que peu de sens ici, puisque notre système ne prédit pas les traits morphologiques qu'elles mesurent.

TABLE 2 – Comparaisons entre les performances de nos modèles (sur différents corpus de test) en utilisant FLAUBERT, MBERT ou pas de représentations contextuelles (no BERT) et les performances de modèles à l'état de l'art.

(a) Résultats pour UD_FRENCH-GSD

Modèle	UPOS	UAS	LAS	CLAS
MBERT	*98,13*	93,95	92,08	88,00
FLAUBERT	**98,56**	95,67	**94,19**	**91,16**
no BERT	97,24	91,74	89,04	84,02
Martin *et al.*	*98,18*	—	92,57	—
Stanza	97,30	91,38	89,05	84,38
UD-Pipe 2	97,98	92,55	90,31	—

(b) Résultats pour FTB-SPRML

Modèle	UPOS	UAS	LAS
MBERT	*98,59*	91,68	88,48
FLAUBERT	**98,78**	**92,56**	**89,64**
no BERT	98,03	88,54	84,54
Le *et al.*	—	91,61	88,47
Constant *et al.*	—	89,19	85,86

(c) Résultats pour UD_FRENCH-SEQUOIA

Modèle	UPOS	UAS	LAS	CLAS
MBERT	*99,01*	94,26	92,75	90,02
FLAUBERT	**99,36**	95,68	**94,40**	92,12
no BERT	97,14	87,84	84,73	79,43
Martin *et al.*	*99,29*	—	*94,20*	—
Stanza	98,19	90,47	88,34	81,77
UD-Pipe 2	*99,32*	94,88	93,81	—

(d) Résultats pour UD_FRENCH-SPOKEN

Modèle	UPOS	UAS	LAS	CLAS
MBERT	*97,19*	83,93	78,13	71,05
FLAUBERT	*96,75*	86,00	80,46	73,97
no BERT	92,62	77,87	69,78	61,03
Martin *et al.*	*97,09*	—	**81,81**	—
Stanza	95,49	75,82	70,71	62,13
UD-Pipe 2	**97,23**	86,27	*81,40*	—

Les scores pour nos modèles sont obtenus en répétant la procédure d'entraînement avec trois germes aléatoires et en conservant les résultats du modèle le plus performant (en terme de LAS) sur le corpus de développement. Ces scores sont calculés par le script d'évaluation officiel de la campagne d'évaluation CoNLL 2018 [4] en considérant les métriques UPOS, UAS et LAS, ainsi que CLAS (Nivre & Fang, 2017) pour les corpus UD. Pour les métriques disponibles dans tout l'état de l'art, nous notons le meilleur résultat **en gras** et les résultats proches (moins de 0,5 points de différence) en *italiques*.

l'apprentissage des structures locales liées aux mots fonctionnels — puisque cette dernière métrique ne tiens précisément pas compte de ces dépendances.

3.2 Résultats de test

Nous présentons dans la table 2, une comparaison de nos résultats sur les jeux de données de test des principaux corpus de français à ceux de l'état de l'art — quand ils sont disponibles.

L'ensemble des modèles auxquels nous nous comparons sont également des variantes de l'algorithme d'analyse à arbre maximal couvrant introduite par Dozat & Manning (2017). Ils diffèrent essentiellement par les représentations lexicales utilisés. Notons toutefois que l'analyseur Stanza (Qi *et al.*, 2020) réalise sa propre segmentation en mots alors que l'analyseur que nous présentons utilise la segmentation proposée par le treebank. Ses résultats ne sont donc pas parfaitement comparables avec les notres et sont donnés à titre indicatif. Tous les modèles utilisent en entrée des embeddings produits par une variante de BERT à l'exception de l'analyseur de Constant *et al.* (2013) qui n'est pas un analyseur à apprentissage profond et de Stanza. UDPipe 2 (Straka *et al.*, 2019) utilise des

représentations BERT issues du modèle MBERT (Devlin *et al.*, 2019), mais contrairement à nous ces auteurs ne les ajustent pas durant l'entraînement de leur analyseur.

Les résultats obtenus par nos modèles, aussi bien pour l'étiquetage morphosyntaxique que pour l'analyse en dépendance sont généralement meilleurs que l'état de l'art, à l'exception des résultats pour UD_FRENCH-SPOKEN. Pour ce dernier corpus, nous conjecturons que les résultats moins bons que l'état de l'art que nous obtenons s'expliquent au mpins en partie par la petite taille et la plus grande irrégularité du corpus d'apprentissage, qui joue en la défaveur de nos modèles : ceux-ci présentant un plus grand nombre de paramètres que les modèles auxquels nous les comparons, leur tendance au surapprentissage est d'autant plus grande. Nous recommandons donc, pour des réutilisations de notre systèmes pour lesquelles les performances sur ce type de données seraient cruciales, une optimisation spécifique des hyperparamètres pour ce corpus afin de compenser cette baisse de performances.

Le tableau 3 donne par ailleurs des estimations des performances de ces modèles à l'inférence en terme de nombre de phrases traitées par seconde et d'occupation mémoire. On constate sans surprise que le passage sur GPU améliore grandement la vitesse, en particulier pour un modèle utilisant FlauBERT, mais que tous ces modèles restent utilisable pour des volumes de données moyens, même sur un ordinateur personnel.

TABLE 3 – Performances de nos modèles à l'inférence. Les valeurs données sont évaluées sur le corpus de test de UD_French-GSD. Les valeurs « CPU » sont mesurées sur un ordinateur portable (processeur Intel Core i7-6500U, 2.50GHz en utilisant deux cœurs) et les valeurs « GPU » sur une machine dédiée au calcul (GPU NVidia GeForce RTX 3090). Les incertitudes sont calculées sur 10 répétitions

Modèle	Vitesse CPU (phrases/s)	Vitesse GPU (phrases/s)	Mémoire vive (GiB)
FLAUBERT	4,09±0,06	20,32±0,08	5,5
no BERT	22,96±0,12	33,03±0,79	3,5

4 Conclusion

Au final, on observe que les résultats de notre analyseur sont à l'état de l'art pour le français écrit tant l'analyse syntaxique que pour l'étiquetage morphosyntaxique. On obtient ce résultat en adaptant les embeddings contextuels de BERT appris sur un modèle spécifique au français (FLAUBERT). L'apport d'autres représentations comme FastText ou les plongements de caractères semble mineur. Pour le français parlé, il reste manifestement une marge de progression qui mérite un approfondissement.

Parmi les perspectives, nous envisageons de tester cette architecture d'analyse syntaxique sur des langues moins bien dotée, comme l'ancien français, ce qui est un contexte de travail où l'utilisation de modèles de plongements contextuels est beaucoup plus complexe à mettre en oeuvre, le manque de données linguistiques disponible rendant complexe le développement de représentations lexicales telles que celles utilisées ici.

Remerciements

Ce travail bénéficié du soutien du projet ANR Profiterole (PRocessing Old French Instrumented TExts for the Representation Of Language Evolution), projet ANR-16-CE38-0010.

Références

Bojanowski P., Grave E., Joulin A. & Mikolov T. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, **5**, 135–146. doi : 10.1162/tacl_a_00051.

Bonfante G., Guillaume B. & Perrier G. (2018). *Application of Graph Rewriting to Natural Language Processing*, volume 1. ISTE Wiley.

Candito M. & Seddah D. (2012). Le corpus Sequoia : annotation syntaxique et exploitation pour l'adaptation d'analyseur par pont lexical. In *Actes de la conférence conjointe JEP-TALN-RECITAL 2012*, volume 2 : Association pour le Traitement Automatique des Langues.

Chu Y.-J. & Liu T.-H. (1965). On the shortest arborescence of a directed graph. *Scientia Sinica*, **14**, 1396–1400.

Constant M., Candito M. & Seddah D. (2013). The ligm-alpage architecture for the spmrl 2013 shared task : Multiword expression analysis and dependency parsing. In *Proceedings of the EMNLP Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL 2013) : shared task track*.

Devlin J., Chang M.-W., Lee K. & Toutanova K. (2019). BERT : Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota : Association for Computational Linguistics.

Dozat T. & Manning C. D. (2017). Deep biaffine attention for neural dependency parsing. In *Proceedings of ICLR*.

Edmonds J. (1967). Optimum branchings. *Journal of Research of the National Bureau of Standards*, **71B**(4), 233. doi : 10.6028/jres.071b.032.

Gerdes K. & Kahane S. (2017). Trois schémas d'annotation syntaxique en dépendance pour un même corpus de français oral : le cas de la macrosyntaxe. In *Actes de l'atelier sur les corpus annotés du français*.

Guillaume B., de Marneffe M.-C. & Perrier G. (2019). Conversion et améliorations de corpus du français annotés en Universal Dependencies. *Traitement Automatique des Langues*, **60**(2), 71.

Hochreiter S. & Schmidhuber J. (1997). Long short-term memory. *Neural computation*, **9**(8), 1735–1780.

Kingma D. P. & Ba J. (2014). Adam : A Method for Stochastic Optimization. In *International Conference on Learning Representations*.

Lacheret A., Kahane S., Beliao J., Dister A., Gerdes K., Goldman J.-P., Obin N., Pietrandrea P. & Tchobanov A. (2014). Rhapsodie : a Prosodic-Syntactic Treebank for Spoken French. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation* : European Language Resource Association. hal : hal-00968959.

Le H., Vial L., Frej J., Segonne V., Coavoux M., Lecouteux B., Allauzen A., Crabbé B., Besacier L. & Schwab D. (2020). FlauBERT : Unsupervised language model pre-training for French. In *Proceedings of the 12th Language Resources and Evaluation Conference*, p. 2479–2490, Marseille, France : European Language Resources Association.

Martin L., Muller B., Ortiz Suárez P. J., Dupont Y., Romary L., de la Clergerie É., Seddah D. & Sagot B. (2020). CamemBERT : a tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, p. 7203–7219 : Association for Computational Linguistics.

Nivre J. & Fang C.-T. (2017). Universal Dependency Evaluation. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, p. 86–95 : Association for Computational Linguistics.

Qi P., Zhang Y., Zhang Y., Bolton J. & Manning C. D. (2020). Stanza : A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics : System Demonstrations*.

Seddah D., Tsarfaty R., Kübler S., Candito M., Choi J. D., Farkas R., Foster J., Goenaga I., Gojenola Galletebeitia K., Goldberg Y., Green S., Habash N., Kuhlmann M., Maier W., Nivre J., Przepiórkowski A., Roth R., Seeker W., Versley Y., Vincze V., Woliński M., Wróblewska A. & Villemonte de la Clergerie E. (2013). Overview of the SPMRL 2013 Shared Task : A Cross-Framework Evaluation of Parsing Morphologically Rich Languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, p. 146–182 : Association for Computational Linguistics.

Straka M., Straková J. & Hajic J. (2019). Evaluating contextualized embeddings on 54 languages in POS tagging, lemmatization and dependency parsing. *CoRR*, **abs/1908.07448**.

Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L. & Gomez A. (2017). Attention is all you need. *Advances in neural information processing systems*, p. 5998–6008.

Wolf T., Debut L., Sanh V., Chaumond J., Delangue C., Moi A., Cistac P., Rault T., Louf R., Funtowicz M., Davison J., Shleifer S., von Platen P., Ma C., Jernite Y., Plu J., Xu C., Le Scao T., Gugger S., Drame M., Lhoest Q. & Rush A. (2020). Transformers : State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing : System Demonstrations*, p. 38–45 : Association for Computational Linguistics. doi : 10.18653/v1/2020.emnlp-demos.6.

Zeman D. *et al.* (2020). Universal dependencies 2.7. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.