Sur l'impact des contraintes structurelles pour l'analyse en dépendances profondes fondée sur les graphes

Caio Corro

Actes de la 6e conférence conjointe Journées d'Études sur la Parole (JEP, 31e édition),
Traitement Automatique des Langues Naturelles (TALN, 27e édition),
Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RÉCITAL, 22e édition)
Nancy, France, 08-19 juin 2020

Sur l'impact des contraintes structurelles pour l'analyse en dépendances profondes fondée sur les graphes

Caio Corro
Université Paris-Saclay, CNRS, LIMSI, 91400, Orsay, France
caio.corro@limsi.fr

RÉSUMÉ

Les algorithmes existants pour l'analyse en dépendances profondes fondée sur les graphes capables de garantir la connexité des structures produites ne couvrent pas les corpus du français. Nous proposons un nouvel algorithme qui couvre l'ensemble des structures possibles. Nous nous évaluons sur les corpus français FTB et Sequoia et observons un compromis entre la production de structures valides et la qualité des analyses.

ABSTRACT

On the impact of structural constraints for graph-based deep dependency parsing

Existing approaches for graph-based deep dependency parsing that force connexity in predicted graphs do no cover the structures observed in French treebanks. We propose a novel algorithm that covers the full set of possible structures. We evaluate our approach on the French corpora FTB and Sequoia and observe a trade-off between the validity of predicted structures and the quality of predictions.

MOTS-CLÉS : analyse syntaxique, analyse en dépendances profondes, optimisation combinatoire.

KEYWORDS: syntactic analysis, deep dependency parsing, combinatorial optimization.

1 Introduction

La structure en dépendances *surfaciques* d'une phrase est usuellement représentée sous forme d'un graphe dirigé où chaque mot est représenté par un nœud et chaque dépendance bi-lexicale par un arc (figure 1). Un nœud supplémentaire est introduit pour connecter le mot racine de la phrase. Le graphe ainsi construit est une arborescence couvrante : chaque nœud sauf la racine possède exactement un arc entrant et il est existe un chemin à partir de la racine jusque chaque nœud. Il en découle qu'une arborescence ne peut pas contenir de cycle. L'analyse en dépendances fondée sur les graphes d'une phrase comporte deux étapes (McDonald *et al.*, 2005) :

1. la construction d'un graphe dirigé et pondéré complet où le poids associé à chaque arc correspond à la vraisemblance d'une relation bi-lexicale calculée par un modèle statistique ;
2. la calcul de l'arborescence couvrante de poids maximal de ce graphe, c'est-à-dire la sélection d'un sous-graphe contraint de poids maximal.

Le sous-graphe calculé peut alors être directement transposé à la structure syntaxique correspondante. Le problème à résoudre lors de la seconde étape a une complexité quadratique par rapport au nombre de mots dans la phrase d'entrée et peut être résolu en utilisant la variante de Fredman & Tarjan (1987)

Volume 2 : Traitement Automatique des Langues Naturelles, pages 197–204.
HAL : hal-02784768.

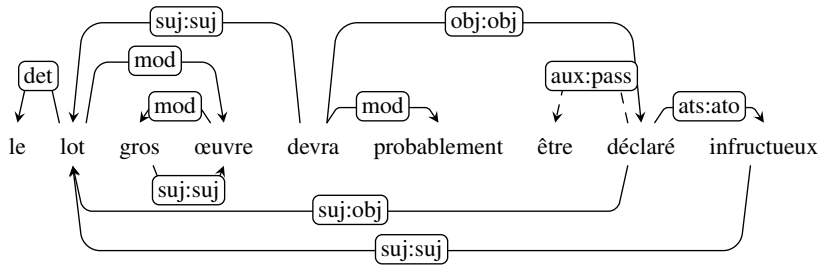

FIGURE 1 – Exemple d'analyse en dépendances surfaciques (arcs au-dessus de la phrase) et en dépendances profondes (combinaison des arcs pleins du dessus et des arcs en dessous) repris de Candito *et al.* (2014). Le mot "être" n'est pas présent dans l'analyse en dépendances profondes et il existe un cycle contenant "gros" et "œuvre".

Complexité	FTB	Sequoia
$O(n^3)$	57.88%	65.34%
$O(n^4)$	83.63%	86.99%
$O(n^5)$	84.34%	87.45%
NP-diff.	100%	100%

TABLE 1 – Couverture en terme de graphes complets du FTB et de Sequoia en fonction de la complexité temporelle de l'analyseur. Le parser en $\mathcal{O}(n^3)$ est celui proposé par Kuhlmann & Jonsson (2015), ceux en $O(n^4)$ et $O(n^5)$ sont ceux proposés par Cao *et al.* (2017), le NP-difficile celui proposé dans cet article.

de l'algorithme de Chu-Liu-Edmonds (Edmonds, 1967). D'autres contraintes sur le sous-graphe peuvent être ajoutées comme la projectivité (Eisner & Satta, 1999) ou le degré de bloc limité et la bonne imbrication (Gómez-Rodríguez *et al.*, 2009; Corro *et al.*, 2016) au prix d'une complexité temporelle plus élevée, voire d'une complexité rédhibitoire.

Nous nous intéressons ici à l'analyse en dépendances *profondes* fondée sur les graphes (figure 1). À la différence des dépendances surfaciques, une structure en dépendances profondes s'abstrait des variations syntaxiques pour représenter plus explicitement les relations prédicats-arguments, entre autres. Cette représentation ne contient que les mots sémantiquement pleins. Nous renvoyons le lecteur au schéma d'annotation du corpus Sequoia (Perrier *et al.*, 2014) pour de plus amples informations. Notons qu'une analyse en dépendances profondes n'est pas une extension des dépendances surfaciques : par exemple sur la figure 1, la dépendance entre « être » et « déclaré » n'est pas présente. Pour l'analyse fondée sur les graphes, le calcul du sous-graphe de poids maximal doit donc prendre en compte les spécificités de cette représentation :

1. le sous-graphe n'est pas contraint d'être couvrant car les mots sémantiquement vides n'apparaissent pas dans la structure syntaxique ;
2. un nœud peut avoir 0 ou plusieurs arcs entrants ;
3. le sous-graphe peut contenir des cycles.

En d'autres termes, la seule propriété qui doit être contrainte sur le sous-graphe pour qu'il soit valide est sa faible connexité, c'est à dire qu'il doit exister un chemin à partir de la racine jusque tous les autres mots nœud du graphe sans prendre en considération la direction des arcs. Malheureusement, ce problème est NP-difficile (Ideker *et al.*, 2002). Bien qu'il existe des algorithmes tractables permettant de calculer des sous-graphes faiblement connexes plus contraints (Kuhlmann & Jonsson, 2015; Cao *et al.*, 2017), ceux-ci ne permettent pas de couvrir l'ensemble des structures présentes dans les jeux de données (voir table 1).

Dans cet article, nous proposons un algorithme qui garantit la connexité des analyses produites tout en couvrant l'ensemble des corpus et nous examinons empiriquement si cette contrainte est importante pour l'analyse en dépendances profondes fondée sur les graphes. En effet, dans le cadre de l'analyse en dépendances surfaciques, Zhang *et al.* (2017) ont montré qu'il était possible de supprimer les contraintes d'arborescence lors de la prédiction, c'est-à-dire de choisir localement un arc entrant pour chaque nœud sans garantir explicitement la contrainte globale d'arborescence. De plus, empiriquement la plupart des graphes produits par leur analyseur non contraint sont des

arborescences. Dans le cas de l'analyse en dépendances profondes, la suppression de la contrainte de connexité transforme le calcul du sous-graphe de poids maximal en la simple sélection de tous les arcs de poids positifs. Cette approche non contrainte est celle choisie par les analyseurs à l'état de l'art pour les dépendances profondes du français. En effet, si l'analyseur de Ribeyre *et al.* (2016) ne se limite pas à une simple sélection des arcs de poids positifs, c'est seulement dû à l'utilisation d'un modèle d'ordre supérieur [1]. Cependant, la connexité du graphe produit n'est pas garantie.

Nos contributions peuvent être résumées comme suit :

1. Nous proposons un nouvel algorithme pour l'analyse en syntaxe profonde qui permet de couvrir l'ensemble des structures des jeux de données tout en garantissant la connexité des graphes produits ;
2. Nous comparons expérimentalement les résultats produits par notre analyseur contraint et la simple sélection des arcs de poids positifs.

L'implémentation pour reproduire les expériences de cet article est disponible en ligne [2].

2 Analyse en dépendances profondes fondée sur les graphes

Nous proposons de réduire le problème de l'analyse syntaxique profonde au problème du sous-graphe connexe de poids maximal (SGCPM), un problème de graphe NP-difficile ayant des applications en chimie et en biologie moléculaire (Dittrich *et al.*, 2008; Ideker *et al.*, 2002; Loboda *et al.*, 2016). À la différence de l'approche proposée par Ribeyre *et al.* (2016), nous contraignons la bonne formation de la structure syntaxique. À la différence des approches proposées par Kuhlmann & Jonsson (2015) et Cao *et al.* (2017), nous pouvons couvrir l'ensemble des jeux de données (voir table 1).

2.1 Réduction à un problème de graphe non dirigé

Soit $\boldsymbol{s} = s_0 ... s_n$ une phrase de n mots où s_0 est un faux mot supplémentaire. Nous construisons un graphe dirigé complet pondéré $G = \langle V, A, \boldsymbol{w} \rangle$ avec $V = \{0...n\}$ l'ensemble des nœuds, $A = V \times V$ l'ensemble des arcs et $\boldsymbol{w} \in \mathbb{R}^{|A|}$ un vecteur de poids indexé par les arcs. Le problème du SGCPM sur un graphe dirigé peut se réduire à sa variante non dirigée en construisant un graphe non dirigé pondéré $G' = \langle V, E, \boldsymbol{w}' \rangle$ avec $E \subset V \times V$ l'ensemble des arêtes et $\boldsymbol{w}' \in \mathbb{R}^{|E|}$ de la façon suivante : pour chaque couple de nœuds $u, v \in V$ tels que $u < v$, nous ajoutons une arêtes (u, v) au graphe G' de poids : $w'_{uv} = max(w_{uv}, w_{vu}, w_{uv} + w_{vu})$. Une solution du SGCPM sur G peut alors être reconstruite à partir d'une solution du SGCPM sur G' en tenant compte des arcs de G qui ont contribué au poids des arêtes de G'.

2.2 Programme mathématique

Nous proposons ici une formulation sous forme de programme mathématique du problème du SGCPM, fondée sur les formulations de Haouari *et al.* (2013) et Loboda *et al.* (2016). Le programme

1. Dans un modèle du première ordre, le poids d'un graphe est la somme des poids des arcs qu'il contient. Les modèles d'ordre supérieur incluent aussi des poids pour les couples d'arcs.
2. https://github.com/FilippoC/deep-syntactic-dependency-parsing-release

que nous proposons est (légèrement) plus simple car il existe un nœud obligatoirement présent dans le sous-graphe : la racine. Le sous-graphe recherché étant connexe, tous les nœuds doivent être accessibles en suivant un chemin à partir de celle-ci : nous devons garantir l'existence d'au moins une traversée possible à partir de la racine. Pour cela, nous représentons l'ensemble des chemins de cette traversée par une arborescence où un arc indique que le nœud de départ est visité avant le nœud d'arrivée dans la traversée [3]. Pour forcer la structure d'arborescence, nous utilisons une variable entière qui représente la distance de la racine à chaque nœud du sous-graphe plutôt qu'une modélisation par flot.

Nous introduisons 3 ensembles de variables binaires. Les variables y_e, $e \in E$, et x_v, $v \in V \setminus \{0\}$, représentent la sélection (valeur 1) ou non (valeur 0) des différentes arêtes et nœuds, respectivement. Les variables z_a, $a \in A$, représentent la sélection des arcs de la traversée utilisée pour garantir la connexité du sous-graphe produit. De plus, les variables entières $d_v, v \in V$, sont utilisées pour décrire les solutions faisables de l'arborescence. Nous notons $\delta(v)$ l'ensemble des arêtes incidentes au nœud v. Le programme mathématique pour résoudre le problème du SGCPM sur G' est défini de la façon suivante :

$$\max \quad \sum_{e \in E} w'_e y_e, \tag{1}$$

$$\text{t.q.} \quad y_e \leq x_v, \qquad \forall v \in V, e \in \delta(v), \tag{2}$$

$$z_{uv} + z_{vu} \leq y_{uv}, \qquad \forall (u,v) \in E, \tag{3}$$

$$\sum_{(u,v) \in A} z_{uv} = x_v, \qquad \forall v \in V \setminus \{0\}, \tag{4}$$

$$d_0 = 1, \tag{5}$$

$$d_v z_{uv} = (d_u + 1) z_{uv}, \qquad \forall (u,v) \in A, \tag{6}$$

$$\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \{0,1\}, \boldsymbol{d} \in [1...n+1]. \tag{7}$$

La contrainte (2) garantit qu'une arête ne peut être sélectionnée que si ses deux nœuds incidents sont sélectionnés. La contrainte (3) s'assure que les arcs qui forment l'arborescence ne peuvent être présents que si l'arête équivalente dans le sous-graphe non-dirigé est présente. Ensuite, les contraintes (4)-(6) assurent la bonne formation de l'arborescence : chaque nœud doit avoir exactement un arc entrant s'il est sélectionné (4) ; la distance de la racine à elle même est égale à un (5) ; si un arc est présent dans l'arborescence, alors la distance de la racine de son nœud de destination est égale à la distance de la racine du nœud de départ plus un (6) [4]. Pour une démonstration de l'exactitude de cette formulation, nous reportons le lecteur à Haouari *et al.* (2013). Ce programme mathématique n'est pas un programme linéaire en nombre entiers à cause de la contrainte (6). Cependant, celle-ci peut être linéarisée en suivant la procédure décrite par Loboda *et al.* (2016). En pratique, nous pouvons donc résoudre ce programme mathématique avec l'outil CPLEX [5].

3. Précisons bien que les arcs de cette arborescence ne représentent pas forcément des dépendances syntaxiques. Les dépendances sélectionnées sont représentées par les arêtes (non-dirigées) du sous-graphe connexe. De plus ils ne décrivent qu'une traversée possible, il peut en exister d'autres.

4. Cette contrainte interdit les cycles dans l'arborescence.

5. https://www.ibm.com/fr-fr/analytics/cplex-optimizer

	% de struct. con.	LP	LR	LF	Exacte
	FULLYSUP				
POSITIVEARCS	64.84 / 66.83	85.90 / 86.15	82.70 / 83.00	**84.27 / 84.55**	17.95 / 18.83
→ FTB	63.16 / 65.39	85.63 / 85.89	82.43 / 82.75	84.00 / 84.29	15.39 / 16.26
→ Sequoia	75.50	88.21	85.02	86.58	34.25
SGCPM	100 / 100	84.70 / 85.02	83.48 / 83.78	84.09 / 84.40	**18.40 / 19.29**
→ FTB	100 / 100	84.43 / 84.77	83.22 / 83.53	83.82 / 84.14	15.82 / 16.72
→ Sequoia	100	87.05	85.77	86.41	34.75
	PRETRAINED				
POSITIVEARCS	69.16 / 70.65	89.10 / 89.46	85.38 / 85.77	**87.20 / 87.57**	21.32 / 22.36
→ FTB	67.41 / 69.05	88.83 / 89.20	85.07 / 85.48	86.91 / 87.30	18.93 / 20.01
→ Sequoia	80.25	91.48	88.02	89.72	36.50
SGCPM	100 / 100	88.19 / 88.59	86.04 / 86.41	87.10 / 87.49	**22.34 / 23.43**
→ FTB	100 / 100	87.89 / 88.32	85.74 / 86.14	86.80 / 87.21	19.83 / 20.97
→ Sequoia	100	90.78	88.61	89.68	38.25

TABLE 2 – Résultats de nos deux analyseurs avec les deux réseaux de neurones en terme de pourcentage de structures connexes, de précision étiquetée, rappel étiqueté, f-mesure étiquetée et de correspondance exacte. Le FTB contenant des erreurs de conversion induisant des structures non connexes, nous reportons deux scores : score sur tout le jeu de test / score sur les phrases ayant des structures de référence connexes seulement.

3 Pondération neuronale

Nous utilisons l'architecture neuronale de Dozat & Manning (2017) avec les mêmes hyper-paramètres. Nous expérimentons avec deux variantes qui diffèrent par la façon dont elles construisent les plongements lexicaux.

L'architecture FULLYSUP est entièrement initialisée aléatoirement et est entrainée de bout en bout. Elle est fondée sur des plongements lexicaux et des plongements de caractères.

L'architecture PRETRAINED utilise un *self-attentive encoder* (Vaswani *et al.*, 2017) pré-entrainé pour extraire les plongements sensibles au contexte (Martin *et al.*, 2019). Étant donné que ce modèle pré-entrainé utilise son propre tokenizer de mots, nous utilisons la sortie qui correspond au premier sous-token de chaque mot. De plus, plutôt que d'utiliser la représentation en sortie de la dernière couche d'attention, nous apprenons une combinaison convexe des couches 4 à 7, de la même façon que proposée pour l'architecture Elmo (Peters *et al.*, 2018). Ensuite, nous concaténons cette sortie à un plongement lexical appris de bout en bout, suivi de la même architecture que Dozat & Manning (2017).

Le calcul de la fonction de perte et de sa dérivée est une étape coûteuse en prédiction structurée. Nous décomposons la fonction de perte par partie, c'est-à-dire par arc, d'une façon similaire à ce qui a été proposé par Dozat & Manning (2017) et Zhang *et al.* (2017) mais adapté aux dépendances profondes : pour chaque arc dans le graphe complet nous utilisons une perte de type entropie croisée binaire.

4 Expériences

Nous nous évaluons sur les corpus French Treebank (FTB, Abeillé *et al.*, 2003) et Sequoia (Candito & Seddah, 2012) convertis en dépendances profondes (Perrier *et al.*, 2014). La division du corpus

pour l'entrainement / la validation / le test comporte 14759 / 1235 / 2541 et 2202 / 497 / 400 phrases pour le FTB et Sequoia, respectivement. Pour l'entrainement, nous concaténons les deux corpus. Le corpus FTB contient des structures non connexes (137 phrases sur 2541 dans le test) qui semblent dues à des erreurs de conversion automatique. Nous reportons donc les résultats à la fois sur le corpus de test complet et sur la sous-partie des phrases dont la structure syntaxique profonde de référence est connexe. De plus, nous utilisons deux analyseurs différents : POSITIVEARCS est un analyseur non structuré qui sélectionne tous les arcs de poids positifs, SGCPM est notre analyseur qui ne produit que des analyses valides, c'est-à-dire que des graphes faiblement connexes. Notons que nos résultats sont dans les mêmes ordres de grandeur que ceux de Ribeyre *et al.* (2016) qui obtiennent une f-mesure de 80.79 sans extraction de charactéristiques supplémentaires et 85.18 en utilisant des charactéristiques extérieurs (analyse en constituants, ...) [6].

La première question qui nous intéresse est de savoir si les réseaux de neurones peuvent apprendre implicitement à ne produire que des structures connexes ou s'il est important de forcer cette contrainte lors de la prédiction. Nous reportons le nombre de structures connexes produites par POSITIVEARCS dans la table 2 (partie de gauche). Notons que les structures non connexes peuvent être problématiques lors de l'utilisation des prédictions pour des tâches cibles.

La seconde interrogation porte sur la qualité des prédictions : est ce que garantir des structures connexes lors de la prédiction permet de corriger des erreurs ? Les scores de nos deux réseaux de neurones sont reportés sur le Tableau 2 (partie de droite). Comme nous pouvons l'observer, forcer les structures à être connexe diminue le score en f-mesure, mais augmente le nombre de phrases dont la structure syntaxique est exactement prédite.

5 Conclusion

Dans cet article, nous proposons un nouvel algorithme pour l'analyse en dépendances profondes fondée sur les graphes qui permet de couvrir l'ensemble des jeux de données du français. Nous avons évalué notre approche sur le français. Nous observons que garantir ces structures impacte négativement les résultats des prédictions. Cependant, sans cet algorithme une partie des structures produites sont non connexes, ce qui peut poser problème pour des tâches cibles. De plus, le nombre de structures parfaitement prédites augmente lorsque l'on force la connexité.

Remerciements

Nous remercions les 3 relecteurices anonymes pour leurs remarques et suggestions. Nous remercions vivement Marie Candito et Djamé Seddah pour nous avoir aidé avec la mise en place du corpus Sequoia et FTB en dépendances profondes ainsi que pour avoir répondu à nos interrogations. Nous remercions Corentin Ribeyre et Djamé Seddah pour nous avoir aidé à mettre en place l'évaluateur. Nous remercions Maximin Coavoux et Matthieu Labeau pour les relectures.

6. En l'absence d'un archivage par identifiant unique et découpage commun publiquement disponible pour le jeu de données, comme réalisé par exemple pour le corpus Abstract Meaning Representation qui identifie chaque parution de façon unique (LDC2016E25, LDC2014T12, ...), nous ne savons pas à quel point ces résultats sont comparables. Nous supposons que le découpage du FTB de Ribeyre *et al.* (2016) suit celui de la shared task SPMRL comme nous, mais nous ne savons pas exactement si les versions sont comparables (correction des dépendances non-locales, expressions multi-mots...).

Références

ABEILLÉ A., CLÉMENT L. & TOUSSENEL F. (2003). Building a treebank for french. In *Treebanks*, p. 165–187. Springer.

CANDITO M., PERRIER G., GUILLAUME B., RIBEYRE C., FORT K., SEDDAH D. & DE LA CLERGERIE É. (2014). Deep syntax annotation of the sequoia French treebank. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, p. 2298–2305, Reykjavik, Iceland : European Language Resources Association (ELRA).

CANDITO M. & SEDDAH D. (2012). Le corpus sequoia : annotation syntaxique et exploitation pour l'adaptation d'analyseur par pont lexical. In *Proceedings of TALN 2012*.

CAO J., HUANG S., SUN W. & WAN X. (2017). Parsing to 1-endpoint-crossing, pagenumber-2 graphs. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 2110–2120, Vancouver, Canada : Association for Computational Linguistics. DOI : 10.18653/v1/P17-1193.

CORRO C., LE ROUX J., LACROIX M., ROZENKNOP A. & WOLFLER CALVO R. (2016). Dependency parsing with bounded block degree and well-nestedness via Lagrangian relaxation and branch-and-bound. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 355–366, Berlin, Germany : Association for Computational Linguistics. DOI : 10.18653/v1/P16-1034.

DITTRICH M. T., KLAU G. W., ROSENWALD A., DANDEKAR T. & MÜLLER T. (2008). Identifying functional modules in protein-protein interaction networks : an integrated exact approach. *Bioinformatics*, **24**(13), i223–i231.

DOZAT T. & MANNING C. D. (2017). Deep biaffine attention for neural dependency parsing. In *Proceedings of the 5th International Conference on Learning Representations*.

EDMONDS J. (1967). Optimum branchings. *Journal of Research of the National Bureau of Standards*, **71**(4), 233–240.

EISNER J. & SATTA G. (1999). Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, p. 457–464 : Association for Computational Linguistics.

FREDMAN M. L. & TARJAN R. E. (1987). Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)*, **34**(3), 596–615.

GÓMEZ-RODRÍGUEZ C., WEIR D. & CARROLL J. (2009). Parsing mildly non-projective dependency structures. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, p. 291–299, Athens, Greece : Association for Computational Linguistics.

HAOUARI M., MACULAN N. & MRAD M. (2013). Enhanced compact models for the connected subgraph problem and for the shortest path problem in digraphs with negative cycles. *Computers & operations research*, **40**(10), 2485–2492.

IDEKER T., OZIER O., SCHWIKOWSKI B. & SIEGEL A. F. (2002). Discovering regulatory and signalling circuits in molecular interaction networks. *Bioinformatics*, **18**(suppl_1), S233–S240.

KUHLMANN M. & JONSSON P. (2015). Parsing to noncrossing dependency graphs. *Transactions of the Association for Computational Linguistics*, **3**, 559–570. DOI : 10.1162/tacl_a_00158.

LOBODA A. A., ARTYOMOV M. N. & SERGUSHICHEV A. A. (2016). Solving generalized maximum-weight connected subgraph problem for network enrichment analysis. In *International Workshop on Algorithms in Bioinformatics*, p. 210–221 : Springer.

MARTIN L., MULLER B., SUÁREZ P. J. O., DUPONT Y., ROMARY L., DE LA CLERGERIE É. V., SEDDAH D. & SAGOT B. (2019). Camembert : a tasty french language model. arXiv preprint : 1911.03894.

MCDONALD R., PEREIRA F., RIBAROV K. & HAJIČ J. (2005). Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, p. 523–530, Vancouver, British Columbia, Canada : Association for Computational Linguistics.

PERRIER G., CANDITO M., GUILLAUME B., RIBEYRE C., FORT K. & SEDDAH D. (2014). Un schéma d'annotation en dépendances syntaxiques profondes pour le français. In *Proc. of TALN 2014*, Marseille, France.

PETERS M., NEUMANN M., IYYER M., GARDNER M., CLARK C., LEE K. & ZETTLEMOYER L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long Papers)*, p. 2227–2237, New Orleans, Louisiana : Association for Computational Linguistics. DOI : 10.18653/v1/N18-1202.

RIBEYRE C., DE LA CLERGERIE E. V. & SEDDAH D. (2016). Accurate deep syntactic parsing of graphs : The case of French. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, p. 3563–3568, Portorož, Slovenia : European Language Resources Association (ELRA).

VASWANI A., SHAZEER N., PARMAR N., USZKOREIT J., JONES L., GOMEZ A. N., KAISER Ł. & POLOSUKHIN I. (2017). Attention is all you need. In *Advances in neural information processing systems*, p. 5998–6008.

ZHANG X., CHENG J. & LAPATA M. (2017). Dependency parsing as head selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics : Volume 1, Long Papers*, p. 665–676, Valencia, Spain : Association for Computational Linguistics.