Tâches auxiliaires pour l'analyse vers graphes de dépendances

Séminaire équipe TALEP

Marie Candito
Laboratoire de Linguistique Formelle,
Université Paris Cité, France

Université Paris Cité

19 mai 2022

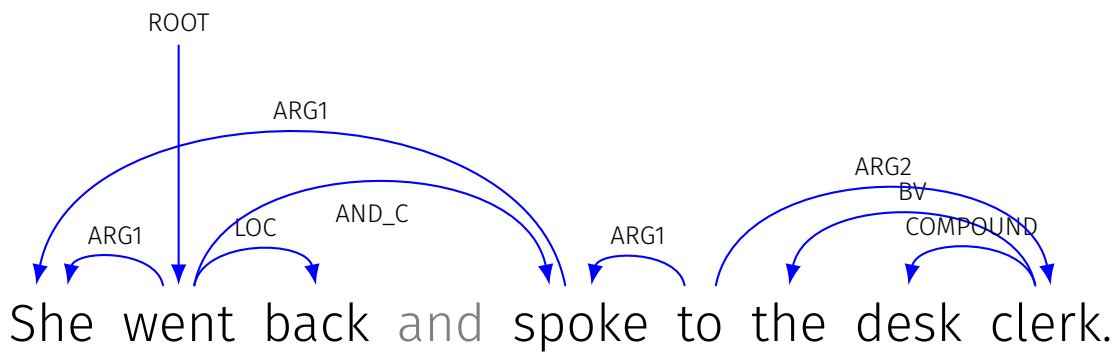
Plan

- Analyse vers graphes de dépendances
  - la tâche
  - les données utilisées
- Principales techniques actuelles
- Quelle gestion de l'**interdépendance** des arcs ?
- Proposition : **tâches auxiliaires**
  - Expériences avec analyseur **biaffine** (à la (Dozat and Manning, 2017, 2018))

Analyse vers graphes de dépendances : la tâche

Sortie cible pour une phrase = graphe de dépendances

- dépendances bilexicales
- allant vers la sémantique
- mais de statut exact variable



Graphe sémantique, données DM, SemEval2015-Task18 (Oepen et al., 2015).

Analyse vers graphes de dépendances : les données

Jeux de données utilisés :

- **EN :** graphes de dépendances **sémantiques**
  - issus de réannotations riches du Penn Treebank (Oepen et al., 2014)
- **FR :** graphes syntaxiques profonds (Ribeyre et al., 2014 ; Candito et al., 2014)
  - données gold : Sequoia (3000 phrases)
  - pseudo-gold : French TreeBank ($\approx$ 20000 phrases)
- Rem : autres jeux couramment utilisés pour le Chinois, Tchèque (Oepen et al., 2014)

EN : graphes sémantiques - Shared tasks Semeval 2014, 2015

”Broad-coverage Semantic Dependency Parsing” (SDP) (Oepen et al., 2014)
Réannotations riches sur le Penn Treebank :

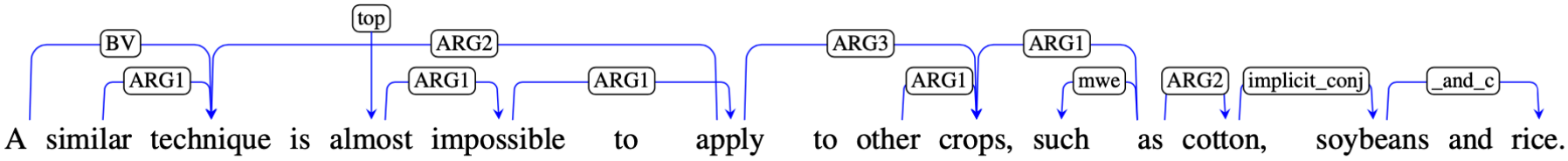
- DM $\rightarrow$ issu du DeepBank (Flickinger et al., 2012)
- PAS : heuristiques transformant PTB en analyses HPSG (Enju HPSG Parser, *https://mynlp.is.s.u-tokyo.ac.jp/enju/index.html*)
- PSD : annotations de type Prague Dependency Bank sur phrases du PTB (Hajič et al., 2012)

EN : graphes sémantiques - Shared tasks Semeval 2014, 2015

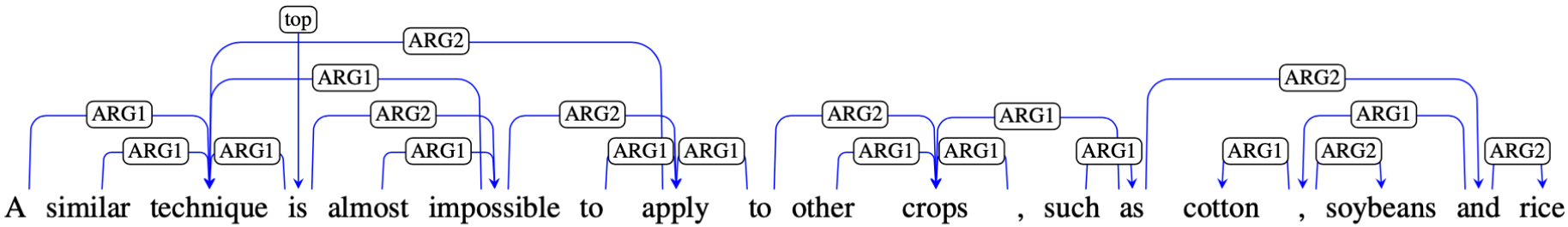
"Broad-coverage Semantic Dependency Parsing" (SDP) (Oepen et al., 2014)

Réannotations riches sur le Penn Treebank :

- DM $\rightarrow$ issu du DeepBank (Flickinger et al., 2012)
- PAS : heuristiques transformant PTB en analyses HPSG (Enju HPSG Parser, *https://mynlp.is.s.u-tokyo.ac.jp/enju/index.html*)
- PSD : annotations de type Prague Dependency Bank sur phrases du PTB (Hajič et al., 2012)



(b) DELPH-IN Minimal Recursion Semantics–derived bi-lexical dependencies (DM).

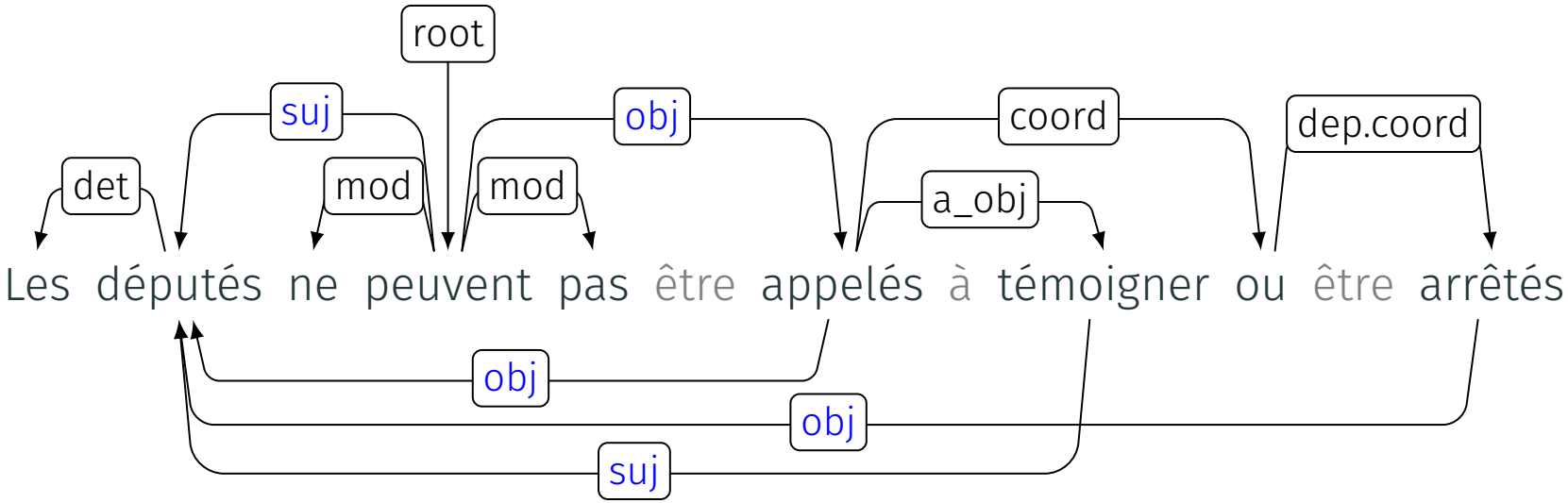


(c) Enju Predicate–Argument Structures (PAS).

(source (Oepen et al., 2014))

FR : graphes syntaxiques profonds

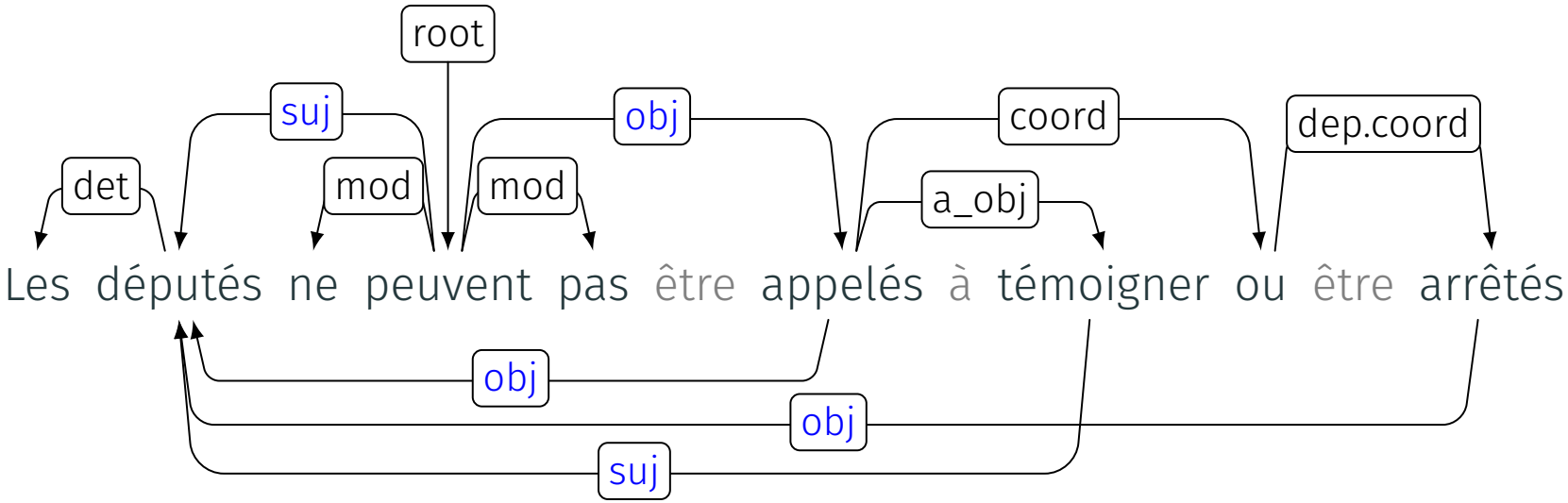
Expliciter plus d'informations déterminées par la syntaxe



- Suppression prépositions régies, complémenteurs, auxiliaires...

FR : graphes syntaxiques profonds

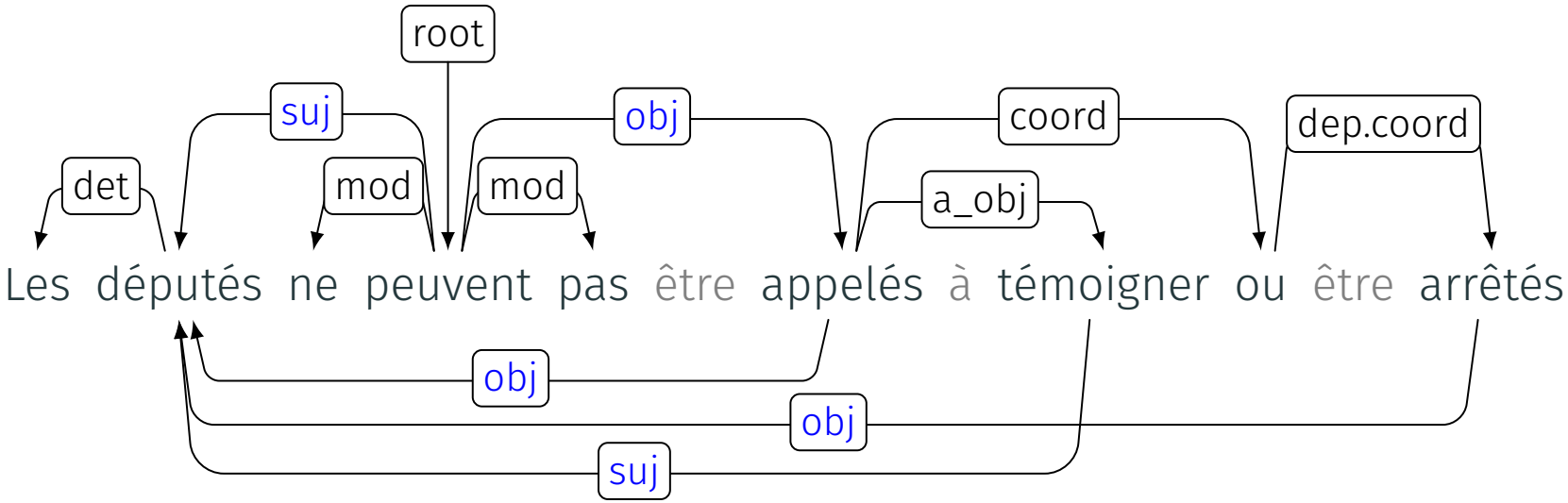
Expliciter plus d'informations déterminées par la syntaxe



- Suppression prépositions régies, complémenteurs, auxiliaires...
- En l'absence de lexique sémantique $\longrightarrow$ étiquettes syntaxiques
- Neutralisation d'alternances $\longrightarrow$ fonctions **canoniques**

FR : graphes syntaxiques profonds

Expliciter plus d'informations déterminées par la syntaxe



- Suppression prépositions régies, complémenteurs, auxiliaires...
- En l'absence de lexique sémantique $\longrightarrow$ étiquettes syntaxiques
- Neutralisation d'alternances $\longrightarrow$ fonctions **canoniques**
- Phénomènes typiques massifs
  - Contrôle/montée : *Cela les$_i$ habitue à être filmés$_i$*
  - Partage de sujet : *Il$_i$ ouvre$_i$ l'enveloppe et la jette$_i$*
  - Passif : Impact déterministe sur le linking des arguments

Analyse vers graphes de dépendances (AGP) : techniques actuelles

Adaptation d'algorithmes de parsing vers **arbres** de dépendances

Améliorations majeures proviennent de l'encodage des tokens

- encodage récurrent (biLSTM)
- représentations contextuelles via modèles de langue préentraînés (type BERT)

Analyse vers graphes de dépendances (AGP) : techniques actuelles

Adaptation d'algorithmes de parsing vers **arbres** de dépendances

Améliorations majeures proviennent de l'encodage des tokens

- encodage récurrent (biLSTM)
- représentations contextuelles via modèles de langue préentraînés (type BERT)

Mais peu d'innovation dans les algorithmes d'analyse eux-mêmes :

- Analyseurs "par transitions" (Nivre, 2003)
- Analyseurs "basés sur les graphes" (McDonald et al., 2005)

AGP : par ”transitions”

Analyseurs ”par transitions” (Nivre, 2003)

Instantiation récente : (Fernández-González and Gómez-Rodríguez, 2020)

- **Configuration** : indice j du mot en cours de traitement
- **Actions possibles à un temps** t :
  - ajouter arc $i \longrightarrow j, \forall i \neq j$
  - ou bien avancer d'un mot dans la phrase

AGP : par ”transitions”

Analyseurs ”par transitions” (Nivre, 2003)

Instantiation récente : (Fernández-González and Gómez-Rodríguez, 2020)

- **Configuration** : indice j du mot en cours de traitement
- **Actions possibles à un temps** t :
  - ajouter arc $i \longrightarrow j, \forall i \neq j$
  - ou bien avancer d'un mot dans la phrase

$\rightarrow$ Simplification extrême du jeu d'actions

$\rightarrow$ Toute la complexité est ramenée dans

- le **scorage des actions** étant donnée une configuration
- l'**encodage** de chaque position j

AGP : par ”transitions”

Analyseurs ”par transitions” (Nivre, 2003)

Instantiation récente : (Fernández-González and Gómez-Rodríguez, 2020)

- **Configuration** : indice j du mot en cours de traitement
- **Actions possibles à un temps** t :
  - ajouter arc $i \longrightarrow j, \forall i \neq j$
  - ou bien avancer d'un mot dans la phrase

$\rightarrow$ Simplification extrême du jeu d'actions

$\rightarrow$ Toute la complexité est ramenée dans

- le **scorage des actions** étant donnée une configuration
- l'**encodage** de chaque position j

- Encodage intégrant les gouverneurs déjà ajoutés pour le mot j

  $\rightarrow$ + Effet (limité) d'interdépendance décisions de rattachement

  $\rightarrow$ - Propagation d'erreurs

AGP : analyseurs ”basés sur les graphes”, ordre 1

Analyseurs ”basés sur les graphes” (McDonald et al., 2005)

- **Ordre 1** : score du graphe/de l’arbre = $\sum$ scores de ses **arcs**

AGP : analyseurs ”basés sur les graphes”, ordre 1

Analyseurs ”basés sur les graphes” (McDonald et al., 2005)

- **Ordre 1** : score du graphe/de l'arbre = $\sum$ scores de ses **arcs**

Instantiation récente : (Dozat and Manning, 2017, 2018)

- Scorage **biaffine** de tous les candidats arcs $i \rightarrow j$:

$$\text{score}(i \rightarrow j) = h_i^{(head)} U h_j^{(dep)} + W(h_i^{(head)} \oplus h_j^{(dep)}) + b$$

AGP : analyseurs "basés sur les graphes", ordre 1

Analyseurs "basés sur les graphes" (McDonald et al., 2005)

- **Ordre 1** : score du graphe/de l'arbre = $\sum$ scores de ses **arcs**

Instantiation récente : (Dozat and Manning, 2017, 2018)

- Scorage **biaffine** de tous les candidats arcs $i \rightarrow j$:

$$\text{score}(i \rightarrow j) = h_i^{(head)} U h_j^{(dep)} + W(h_i^{(head)} \oplus h_j^{(dep)}) + b$$

- Parsing vers arbres : utilisation de la contrainte d'arbre
  - **Softmax** sur scores pour obtenir distribution $P(head = .|dep = j)$
  - **Apprentissage** : entropie croisée $-log(P(head = \text{gold } i|dep = j)$
  - **Inférence** : 1 seule tête par mot, algo de type "Max Spanning Tree"

AGP : analyseurs "basés sur les graphes", ordre 1

Analyseurs "basés sur les graphes" (McDonald et al., 2005)

- **Ordre 1** : score du graphe/de l'arbre = $\sum$ scores de ses **arcs**

Instantiation récente : (Dozat and Manning, 2017, 2018)

- Scorage **biaffine** de tous les candidats arcs $i \rightarrow j$:

  $$\text{score}(i \rightarrow j) = h_i^{(head)} U h_j^{(dep)} + W(h_i^{(head)} \oplus h_j^{(dep)}) + b$$

- Parsing vers arbres : utilisation de la contrainte d'arbre
  - **Softmax** sur scores pour obtenir distribution $P(head = .|dep = j)$
  - **Apprentissage** : entropie croisée $-log(P(head = \text{gold } i|dep = j)$
  - **Inférence** : 1 seule tête par mot, algo de type "Max Spanning Tree"

- Parsing vers graphes :
  - $score(i \rightarrow j)$ transformé en proba d'existence $P(i \rightarrow j \text{ existe}|i, j)$
  - **Apprentissage** : entropie croisée binaire pour tous les couples (i, j)
  - **Inférence** : sans contrainte, prédiction des arcs de proba > 0,5

AGP : analyseurs ”basés sur les graphes”, ordre 2

Analyseurs ”basés sur les graphes” (McDonald et al., 2005)

- **Ordre 2** : décomposition du graphe en ”facteurs” contenant 1, 2, ou 3 arcs

AGP : analyseurs ”basés sur les graphes”, ordre 2

Analyseurs ”basés sur les graphes” (McDonald et al., 2005)

- **Ordre 2** : décomposition du graphe en ”facteurs” contenant 1, 2, ou 3 arcs

Instantiation récente, ordre 2 : (Wang et al., 2019)

- Facteurs à 2 arcs : arcs frères, grand-parents, co-parents ...
- Interdépendance cf. scorage de paires d'arcs
- Au prix d'une inférence approchée en $O(n^3)$

Analyse vers graphes de dépendances : techniques actuelles

Les améliorations majeures actuelles proviennent de l'**encodage** :

- Représentation contextualisée d'une position dans la phrase
  - Remplacement de traits spécifiques par un **encodage récurrent** (biLSTM) (Kiperwasser and Goldberg, 2016)
  - Avec en amont des représentations contextuelles obtenues via **modèles de langue pré-entraînés (type BERT)** (Devlin et al., 2019)

Analyse vers graphes de dépendances : techniques actuelles

Les améliorations majeures actuelles proviennent de l'**encodage** :

- Représentation contextualisée d'une position dans la phrase
  - Remplacement de traits spécifiques par un **encodage récurrent** (biLSTM) (Kiperwasser and Goldberg, 2016)
  - Avec en amont des représentations contextuelles obtenues via **modèles de langue pré-entraînés (type BERT)** (Devlin et al., 2019)
- Recherche automatisée d'architecture neuronale
  - Apprentissage de **combinaison de différentes couches** de représentations contextuelles (Wang et al., 2021)

  $\rightarrow$ état de l'art actuel du parsing vers graphes, données EN

Quelle interdépendance dans la prédiction des arcs ?

Analyseur par transitions :

- Encodage des arcs précédemment prédits

Quelle interdépendance dans la prédiction des arcs ?

Analyseur par transitions :

- Encodage des arcs précédemment prédits

Analyseur biaffine "basé sur les graphes" ordre 1 ($O(n^2)$) :

- Analyseur vers **arbres**
  - $\rightarrow$ utilisation de la contrainte d'arbre
    - à l'apprentissage et à l'inférence

Quelle interdépendance dans la prédiction des arcs ?

Analyseur par transitions :

- Encodage des arcs précédemment prédits

Analyseur biaffine "basé sur les graphes" ordre 1 ($O(n^2)$) :

- Analyseur vers **arbres**
  - $\rightarrow$ utilisation de la contrainte d'arbre
    - à l'apprentissage et à l'inférence

- Analyseur vers **graphes**
  - Aucune contrainte
  - Indépendance totale dans scorage et prédiction des différents arcs

Quelle interdépendance dans la prédiction des arcs ?

Analyseur par transitions :

- Encodage des arcs précédemment prédits

Analyseur biaffine "basé sur les graphes" ordre 1 ($O(n^2)$) :

- Analyseur vers **arbres**
  - $\rightarrow$ utilisation de la contrainte d'arbre
    - à l'apprentissage et à l'inférence
- Analyseur vers **graphes**
  - Aucune contrainte
  - Indépendance totale dans scorage et prédiction des différents arcs

Analyseur "basé sur les graphes" ordre 2 ($O(n^3)$) : (Wang et al., 2019)

- Interdépendance grâce à scorage de paires d'arcs
- Au prix d'une inférence approchée en $O(n^3)$

Analyseur biaffine : analyse d'erreur

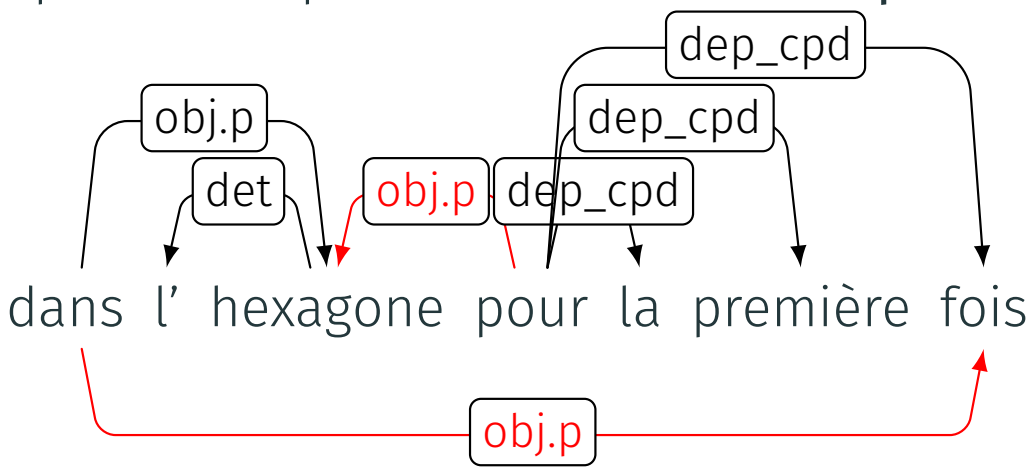
Contexte : Test analyseur biaffine sur graphes synt. profonds FR
$\rightarrow$ Gain par rapport à architecture non neuronale (Ribeyre et al., 2016) :

- base $F_l = 80,9$
- +traits issus d'analyseur riche FrMG (De La Clergerie, 2010) $F_l = 84,9$
- FlauBERT + Parser biaffine $\rightarrow F_l = 86,8$

Analyseur biaffine : analyse d'erreur

Contexte : Test analyseur biaffine sur graphes synt. profonds FR
Mais erreurs parfois grossières !

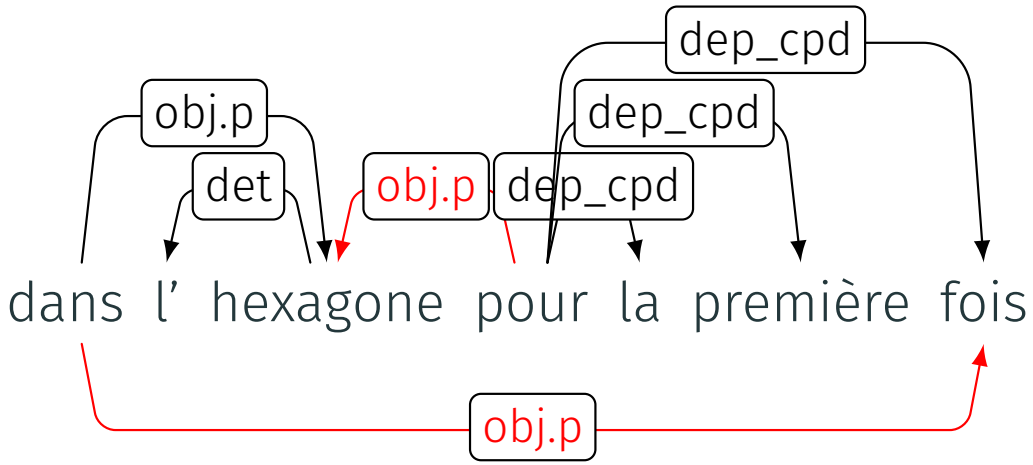
- En particulier prédiction d'**arcs incompatibles**



Analyseur biaffine : analyse d'erreur

Contexte : Test analyseur biaffine sur graphes synt. profonds FR
Mais erreurs parfois grossières !

- En particulier prédiction d'**arcs incompatibles**



- Trop de tokens déconnectés
  - performance repérage tokens 0 arc entrant : P/R/F ≈ 83/94/89
  - plus généralement, accuracy sur nb d'arcs entrants : Acc≈ 92, 2%

Proposition : Tâches auxiliaires concernant plusieurs arcs

Apprentissage multi-tâches avec :

Tâches principales A + L : prédiction des arcs et de leurs étiquettes

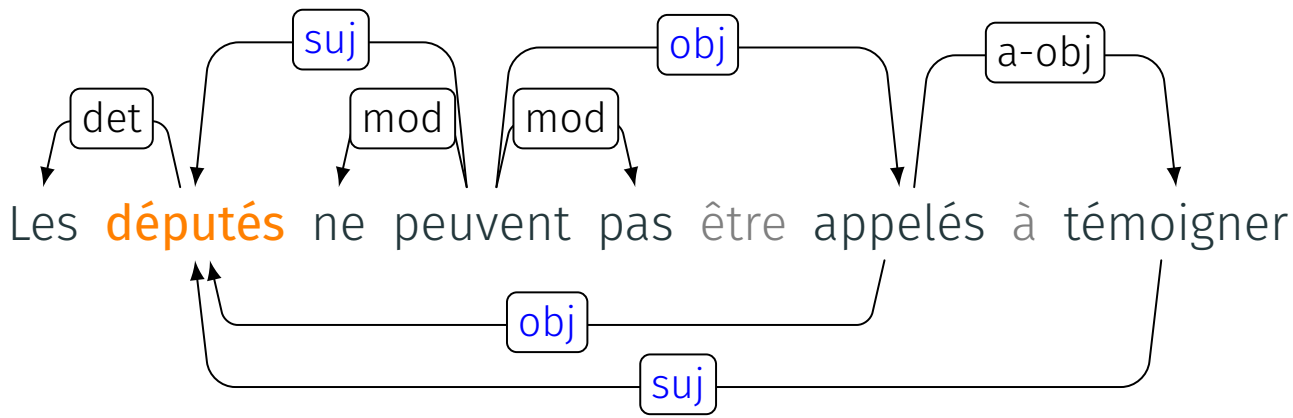
Proposition : Tâches auxiliaires concernant plusieurs arcs

Apprentissage multi-tâches avec :

Tâches principales A + L : prédiction des arcs et de leurs étiquettes

Tâches auxiliaires, pour chaque token :

- H : prédire le nombre de gouverneurs d'un mot
- D : prédire le nombre de dépendants d'un mot



- Par ex. pour députés :
  - H=3, D=1

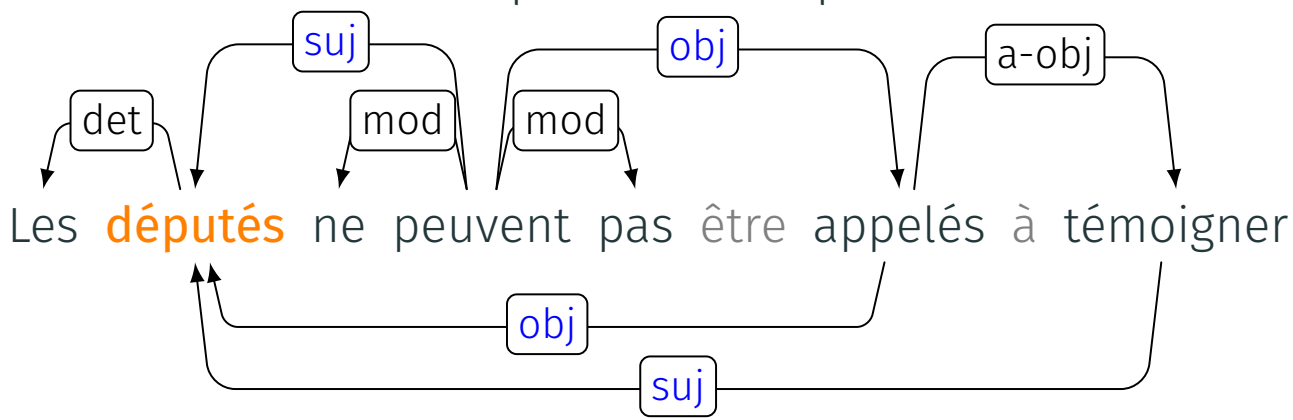
Proposition : Tâches auxiliaires concernant plusieurs arcs

Apprentissage multi-tâches avec :

Tâches principales A + L : prédiction des arcs et de leurs étiquettes

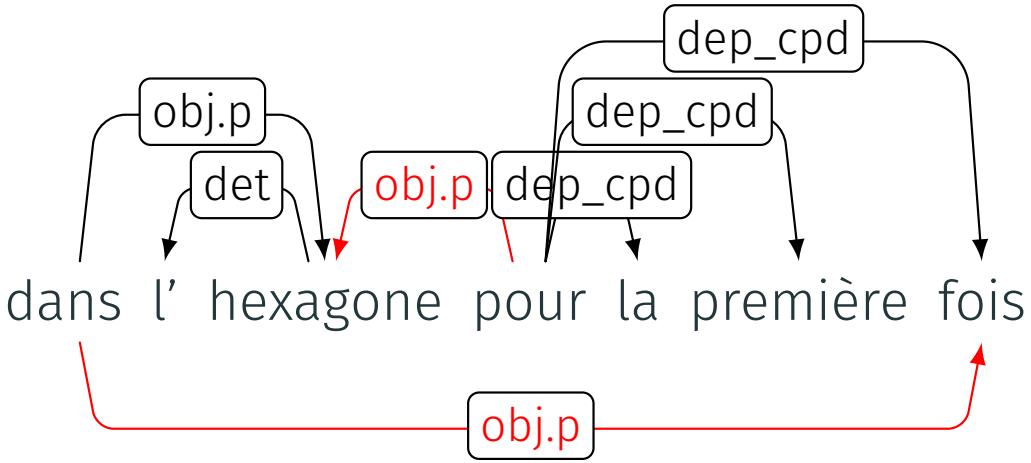
Tâches auxiliaires, pour chaque token :

- H : prédire le nombre de gouverneurs d'un mot
- D : prédire le nombre de dépendants d'un mot
- B/S : prédire l'ensemble des étiquettes des arcs entrants
  - B : sous forme de vecteur "bag of labels" (BOL)
  - S : sous forme d'étiquettes atomiques



- Par ex. pour députés :
  - H=3, D=1
  - S=*obj+suj+suj* (multiensemble, pris atomiquement)

Intuition sous-jacente



Hypothèse par ex. impact tâche H :

- pour token w_j, la représ. récurrente partagée r_j capturerait
- H=0 $\Leftrightarrow$ scores $* \rightarrow j$ faibles
- et inversement

Pour un token w_j composant de composé :

- $\mathbf{r}_j$ optimisée pour aboutir à S='dep_cpd', H=1, D=0
- hypothèse que $\mathbf{r}_j$ utilisée pour les tâches A et L favorisera de ne scorer positivement qu'un seul arc entrant, avec l'étiquette *dep_cpd*.

Apprentissage multi-tâches : partage et spécialisation

Encodage récurrent partagé par toutes les tâches :

$$\begin{aligned}
\mathbf{h}_i^{(\text{bert})} &= (\text{BERT}(w_{1:n}))_i \\
\mathbf{v}_i &= \mathbf{h}_i^{(\text{bert})} \oplus \mathbf{e}_i^{(\text{word})} (\oplus \mathbf{e}_i^{(\text{lemma})} \oplus \mathbf{e}_i^{(\text{POS})}) \\
\mathbf{r}_{1:n} &= \text{biLSTM}(\mathbf{v}_{1:n})
\end{aligned}$$

Tâches A et L :

- Spécialisation des représentations :

$$\begin{aligned}
\mathbf{h}_i^{(\text{arc-head})} &= \text{MLP}^{(\text{arc-head})}(\mathbf{r}_i) & \mathbf{h}_i^{(\text{lab-head})} &= \text{MLP}^{(\text{lab-head})}(\mathbf{r}_i) \\
\mathbf{h}_i^{(\text{arc-dep})} &= \text{MLP}^{(\text{arc-dep})}(\mathbf{r}_i) & \mathbf{h}_i^{(\text{lab-dep})} &= \text{MLP}^{(\text{lab-dep})}(\mathbf{r}_i)
\end{aligned}$$

- Scores des arcs et des étiquettes d'arcs :

$$\begin{aligned}
s_{i \to j}^{(\text{arc})} &= \mathbf{h}_j^{(\text{arc-dep})} \mathbf{U}^{(\text{arc})} \mathbf{h}_i^{(\text{arc-head})\top} + \mathbf{b}^{(\text{arc})} \\
s_{i \to j}^{(\text{l})} &= \mathbf{h}_j^{(\text{lab-dep})} \mathbf{U}^{(\text{l})} \mathbf{h}_i^{(\text{lab-head})\top} + \mathbf{b}^{(\text{l})}
\end{aligned}$$

Apprentissage multi-tâches : partage et spécialisation

Tâches auxiliaires :

- pour chaque token w_j
- partage de la représentation récurrente $\mathbf{r_j}$
- 1 MLP de spécialisation par tâche : $\mathrm{MLP}^{(\mathrm{H})}, \mathrm{MLP}^{(\mathrm{D})}, \mathrm{MLP}^{(\mathrm{B})}, \mathrm{MLP}^{(\mathrm{S})}$

Tâches H et D :

- $nbh_j = \mathrm{MLP}^{(\mathrm{H})}(\mathbf{r}_j)$
- $nbd_j = \mathrm{MLP}^{(\mathrm{D})}(\mathbf{r}_j)$
- perte = erreur au carré moyenne, sur tous les tokens d'une phrase

Tâche S : (étiquettes complexes, e.g. `obj+suj+suj`) :

- score des étiquettes : $\mathrm{MLP}^{(\mathrm{S})}(\mathbf{r}_j)$

Tâche B : prédiction du vecteur bag-of-label, taille = $|L|$

- $\mathbf{BOL}_j = \mathrm{MLP}^{(\mathrm{B})}(\mathbf{r}_j)$
- perte = distance L2 au vecteur BOLD gold, pour chaque token

"Propagation en pile" (Zhang and Weiss, 2016)

(Zhang and Weiss, 2016) : couche cachée d'un tagger utilisée à l'encodage de tokens pour un analyseur en dépendances

Dans notre cas :

- utilisation de la couche cachée des MLP de tâches auxiliaires
- Par exemple, pour utiliser tâche H en mode propagation en pile, on remplace

$$s_{i\to j}^{(\text{arc})} = \mathbf{h}_j^{(\text{arc-dep})} \mathbf{U}^{(\text{arc})} \mathbf{h}_i^{(\text{arc-head})\top} + \mathbf{b}^{(\text{arc})}$$

par

$$s_{i\to j}^{(\text{arc})} = (\mathbf{h}_j^{(\text{arc-dep})} \oplus c^{(\text{H})} \text{hidden}_j^{(\text{H})}) \mathbf{U}^{(\text{arc})} \mathbf{h}_i^{(\text{arc-head})\top} + \mathbf{b}^{(\text{arc})}$$

Protocole : hyperparamètres

2 modes :

- Mode (Flau)BERT comme seule source de paramètres pré-entraînés
  - $\text{BERT}_{\text{base-uncased}}$ (Devlin et al., 2019) et $\text{FlauBERT}_{\text{base-cased}}$ (Le et al., 2020) en mode "**fine-tuning**"
  - ni lemmes, ni POS, $\mathbf{e}_i^{(\text{word})}$ random
  - Reprise d'hyp. dans littérature (biLSTM, $\text{MLP}^{(arc-head)}$, ...)
  - Arrêt précoce, si tous les F_l diminuent sur dev
  - Réglage systématique du taux d'apprentissage
- **Réglage sur graphes synt. profonds FR**
  - combinaison de tâches auxiliaires
  - dropout lexical
- Réutilisation pour tester **avec ou sans propagation en pile**
- **Réutilisation sur graphes sémantiques EN**
- Mode comparaison résultats antérieurs sur graphes EN
  - BERT **figé**, +embeddings de lemmes et POS (gold)

Résultats FR, sur dev

Tâches aux.	Propagation en pile	F_l moyen	écart-type
∅	NA	86,79	0,19
H	non	86,82	0,54
D	non	86,83	0,40
S	non	86,98	0,30
B	non	87,05	0,49
B+H	non	**87,32*****	0,18
D+H	non	86,61	0,71
H+S	non	87,04	0,17
D+H+S	non	86,86	0,39
B+H+S	non	87,14**	0,39
B+D+H+S	non	**87,35*****	0,26
B+H	$c^{(B)}$=1 $c^{(H)}$=1	87,49	0,06
B+H	$c^{(B)}$=1 $c^{(H)}$=10	**87,66**$^{+++}$	0,18

Table 1 – Résultats sur l'ensemble **dev** français, en mode FlauBERT_tuned, pour diverses combinaisons de tâches, avec et sans propagation en pile. Col3-4 : Fscore étiqueté moyen et écart-type (sur 9 lancers). $^{***}/^{**}$: différence significative (test de permutation exact Fisher-Pitman) avec 1ère ligne

Analyse résultats FR, meilleure combinaison B+H

Tâche H = nombre de gouverneurs réalisable

- directement ($\mathrm{MLP}^{(\mathrm{H})}$)
- indirectement au sein des graphes prédits

Config	Préc. ds graphes prédits			Préc. directe	
	B/S	H	D	B	H
aucune tâche aux.	86,7	92,2	89,5	-	-
B+H sans propag	87,9	93,5	90,1	87,6	**96,5**
B+H propag (1,10)	88,4	93,9	90,4	87,6	96,2

Conclusion : les tâches auxiliaires H, B/S semblent sous-employées

Résultats sur graphes anglais

Résultats sur graphes sémantiques EN :

Gains modestes, en- et hors-domaine, mais réguliers

Tâches		DM	PAS	PSD	Moy.		DM	PAS	PSD	Moy.
∅	ID	93,7	93,9	80,7	89,4	OOD	90,3	92,0	79,8	87,4
B+H		94,2	94,3	81,2	89,9		91,0	92,8	80,2	88,0

Table 2 – Fscore étiqueté moyen (sur 9 lancers) sur jeux de test anglais ”en domaine” (ID) et ”hors domaine” (OOD). Résultats B+H signif. plus hauts que ∅ pour DM ID, DM OOD, PAS ID, PAS OOD ($p < 0.001$) et PSD ID ($p < 0.01$).

Résultats sur graphes anglais

Résultats sur graphes sémantiques EN :

Gains modestes, en- et hors-domaine, mais réguliers

Tâches		DM	PAS	PSD	Moy.		DM	PAS	PSD	Moy.
∅	ID	93,7	93,9	80,7	89,4	OOD	90,3	92,0	79,8	87,4
B+H		94,2	94,3	81,2	89,9		91,0	92,8	80,2	88,0

Table 2 – Fscore étiqueté moyen (sur 9 lancers) sur jeux de test anglais "en domaine" (ID) et "hors domaine" (OOD). Résultats B+H signif. plus hauts que ∅ pour DM ID, DM OOD, PAS ID, PAS OOD ($p < 0.001$) et PSD ID ($p < 0.01$).

Mais résultats pas à l'état de l'art :

	ID	OOD
Notre implé Biaffine + BERT + tâches BH (en mode "BERT figé" + lemmes + POS)	90,2	87,9
(Fernández-González and Gómez-Rodríguez, 2020) Transitions + beam5 + char + lemma + POS + BERT	90,7	88,8
(Wang et al., 2021) Biaffine + combi auto couches BERT	**91,7**	**90,2**

Conclusion

- Tâches auxiliaires apportent gains modestes mais systématiques
- Probablement sous-employées
- Technique simple et orthogonale à l'architecture neuronale / l'algorithme d'analyse


Merci

Références i

Candito, M., Perrier, G., Guillaume, B., Ribeyre, C., Fort, K., Seddah, D., and 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)*, pages 2298–2305, Reykjavik, Iceland.

De La Clergerie, É. (2010). Convertir des dérivations TAG en dépendances. In *Actes de la 17e conférence sur le Traitement Automatique des Langues Naturelles. Articles longs*, pages 91–100, Montréal, Canada.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL 2019*, pages 4171–4186, Minneapolis, Minnesota.

Références ii

Dozat, T. and Manning, C. D. (2017). Deep Biaffine Attention for Neural Dependency Parsing. In *5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings*, Toulon, France. OpenReview.net.

Dozat, T. and Manning, C. D. (2018). Simpler but More Accurate Semantic Dependency Parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers)*, pages 484–490, Melbourne, Australia.

Fernández-González, D. and Gómez-Rodríguez, C. (2020). Transition-based Semantic Dependency Parsing with Pointer Networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7035–7046, Online.

Références iii

Flickinger, D., Zhang, Y., and Kordoni, V. (2012). Deepbank : A dynamically annotated treebank of the wall street journal. In *Proceedings of the Eleventh International Workshop on Treebanks and Linguistic Theories (TLT11)*, pages 85–96, Lisbon, Portugal.

Hajič, J., Hajičová, E., Panevová, J., Sgall, P., Bojar, O., Cinková, S., Fučíková, E., Mikulová, M., Pajas, P., Popelka, J., Semecký, J., Šindlerová, J., Štěpánek, J., Toman, J., Urešová, Z., and Žabokrtský, Z. (2012). Announcing Prague Czech-English Dependency Treebank 2.0. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3153–3160, Istanbul, Turkey. European Language Resources Association (ELRA).

Kiperwasser, E. and Goldberg, Y. (2016). Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *Transactions of the Association for Computational Linguistics*, 4 :313–327.

Références iv

Le, H., Vial, L., Frej, J., Segonne, V., Coavoux, M., Lecouteux, B., Allauzen, A., Crabbé, B., Besacier, L., and Schwab, D. (2020). FlauBERT : des modèles de langue contextualisés pré-entraînés pour le français (FlauBERT : Unsupervised Language Model Pre-training for French). In *Actes de JEP-TALN 2020*, pages 268–278, Nancy, France.

McDonald, R., Pereira, F., Ribarov, K., and Hajič, J. (2005). Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *HLT-EMNLP*, pages 523–530, Vancouver, British Columbia, Canada.

Nivre, J. (2003). An efficient algorithm for projective dependency parsing. In *The 8th International Workshop of Parsing Technologies (IWPT 2003)*, Nancy, France.

Références v

Oepen, S., Kuhlmann, M., Miyao, Y., Zeman, D., Cinková, S., Flickinger, D., Hajič, J., and Urešová, Z. (2015). SemEval 2015 Task 18 : Broad-Coverage Semantic Dependency Parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 915–926, Denver, USA.

Oepen, S., Kuhlmann, M., Miyao, Y., Zeman, D., Flickinger, D., Hajič, J., Ivanova, A., and Zhang, Y. (2014). SemEval 2014 Task 8 : Broad-Coverage Semantic Dependency Parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 63–72, Dublin, Ireland.

Ribeyre, C., Candito, M., and Seddah, D. (2014). Semi-Automatic Deep Syntactic Annotations of the French Treebank. In *Proceedings of the 13th International Workshop on Treebanks and Linguistic Theories (TLT13)*, pages 184–197, T”ubingen, Germany.

Références vi

Ribeyre, C., de la Clergerie, E. V., and Seddah, D. (2016). Accurate Deep Syntactic Parsing of Graphs : The Case of French. In *Proceedings of LREC 2016*, pages 3563–3568, Portorož, Slovenia.

Wang, X., Huang, J., and Tu, K. (2019). Second-Order Semantic Dependency Parsing with End-to-End Neural Networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4618, Florence, Italy.

Wang, X., Jiang, Y., Bach, N., Wang, T., Huang, Z., Huang, F., and Tu, K. (2021). Automated concatenation of embeddings for structured prediction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1 : Long Papers)*, pages 2643–2660, Online. Association for Computational Linguistics.

Références vii

Zhang, Y. and Weiss, D. (2016). Stack-propagation : Improved Representation Learning for Syntax. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 1557–1566, Berlin, Germany.