Traitement automatique des langues

Varia

sous la direction de
Emmanuel Morin
Sophie Rosset
Pascale Sébillot

Vol. 59 - n°1 / 2018

ATALA

Varia

ATALA

Revue de
l'Association
pour le Traitement
Automatique
des Langues

TAL
Vol.
59

n°1
2018

Varia

ATALA

Traitement automatique des langues

Revue publiée depuis 1960 par l'Association pour le Traitement Automatique des LAngues (ATALA), avec le concours du CNRS, de l'Université Paris VII et de l'Université de Provence

ISSN 1965-0906

http ://atala.org/revuetal

Traitement automatique des langues

Comité de rédaction

Traitement automatique des langues

Volume 59 – n°1 / 2018

VARIA

Table des matières

Préface

Les activités du comité de rédaction ont été fortement marquées par l'évènement tragique que fut la disparition de notre collègue et amie Isabelle Tellier, qui nous a quittés le 1er juin 2018. Isabelle était rédactrice en chef de la revue TAL et a assumé cette responsabilité tant que ses forces le lui ont permis, suscitant notre admiration pour le courage dont elle a fait preuve jusqu'au bout. Nous conservons tous en tête son sourire et son enthousiasme. Elle a coordonné avec nous ce volume. Nous le lui dédions.

Comme il est de coutume, cette préface de numéro non thématique donne des nouvelles de notre comité de rédaction, mentionne les évolutions dans la gestion de notre revue et fournit des statistiques sur les articles soumis et publiés, avant de présenter brièvement les articles du numéro.

Certains évènements récents nous ont poussés à revoir ou préciser certaines de nos procédures. En particulier, une longueur maximale des articles soumis avait été adoptée dès 2012 mais avait été quelque peu oubliée depuis. Nos conseils aux auteurs mentionnent donc maintenant explicitement une longueur comprise entre 20 et 25 pages dès la phase de soumission.

Un très grand pas en avant a été accompli dans la production automatique des numéros sous forme d'un seul fichier PDF incluant couverture et table des matières. La procédure est maintenant maîtrisée. Ces numéros sont produits par Maxime Amblard et Sophie Rosset.

Accroître la visibilité de la revue et permettre le moissonnage automatique de nos métadonnées par les sites de référencement est un souci constant. Ceci passe par la mise en place d'un dépôt de nos métadonnées visible par les moteurs d'indexation. Philippe Muller et Cécile Fabre se sont investis dans cette tâche.

Pour ce qui est de la régularité de publication, nous pouvons là aussi nous montrer satisfaits de l'établissement d'un calendrier prévisionnel qui nous permet de caler les différents numéros d'un même volume et de tenir de façon plus régulière les réunions du comité de rédaction. Rappelons que l'une des caractéristiques de notre revue, à laquelle nous sommes foncièrement attachés, est la tenue des réunions du comité de rédaction au cours desquelles nous décidons collégialement, à l'appui des relectures reçues, de l'acceptation ou du rejet des articles soumis.

Intitulé	Vol.	N°	Année	Soumis	Acceptés	% acceptés
Varia	56	1	2015	20	4	20,0 %
Sémantique distributionnelle	56	2	2015	7	4	57,1 %
Recherche d'information	56	3	2015	12	3	25,0 %
Sous-total	**56**		**2015**	**39**	**11**	**28,2 %**
Varia	57	1	2016	19	5	26,3 %
TAL et éthique	57	2	2016	7	3	42,9 %
TALP et didactique	57	3	2016	14	5	35,7 %
Sous-total	**57**		**2016**	**40**	**13**	**32,5 %**
Varia	58	1	2017	8	3	37,5 %
Trait. auto. de la langue jur.	58	2	2017	4	2	50,0 %
Trait. auto. de l'arabe et des langues apparentées	58	3	2017	14	4	28,6 %
Sous-total	**58**		**2017**	**26**	**9**	**34,6%**
Varia	59	1	2018	10	3	30,0 %
Total	**Dix derniers n^{os}**			**115**	**36**	**31,3 %**

Tableau 1. *Taux de sélection aux appels de la revue TAL sur les dix derniers numéros de la période 2015-2018*

Intitulé	Vol.	N°	Année	% 1er auteur hors France	% en anglais
Varia	56	1	2015	0,0 %	0,0 %
Sémantique distributionnelle	56	2	2015	0,0 %	0,0 %
Recherche d'information	56	3	2015	0,0 %	0,0 %
Pourcentages par volume	**56**		**2015**	**0,0 %**	**0,0 %**
Varia	57	1	2016	20,0 %	20,0 %
TAL et éthique	57	2	2016	0,0 %	0,0 %
TALP et didactique	57	3	2016	80,0 %	40,0 %
Pourcentages par volume	**57**		**2016**	**33,3 %**	**20,0 %**
Varia	58	1	2017	33,3 %	33,3 %
Trait. auto. de la langue jur.	58	2	2017	50,0 %	50,0 %
Trait. auto. de l'arabe et des langues apparentées	58	3	2017	100,0 %	50,0 %
Pourcentages par volume	**58**		**2017**	**61,1 %**	**44,4 %**
Varia	59	1	2018	0,0 %	0,0 %
Pourcentages totaux	**Dix derniers n^{os}**			**23,6 %**	**16,1 %**

Tableau 2. *Proportion des articles publiés d'un premier auteur hors de France et proportion des articles publiés rédigés en anglais sur les dix derniers numéros de la période 2015-2018. Attention, les pourcentages totaux ne sont pas de simples moyennes des chiffres donnés plus haut, car les dénominateurs changent.*

Passons maintenant à nos statistiques. Elles considèrent toujours les dix derniers numéros sur les trois dernières années, en l'occurrence donc, du début de 2015 jusqu'à ce numéro *Varia* de 2018 inclus. Le tableau 1 donne les taux de sélection par numéro et par volume. La ligne du total synthétise ces chiffres sur l'ensemble des dix numéros considérés.

Le taux de sélection sur l'ensemble de ces numéros s'élève à 31,3 % en moyenne, c'est-à-dire que, sur trois articles soumis, un est accepté. Ce taux est stable dans le temps, entre 31 et 35 %, d'après les chiffres donnés depuis le numéro 51-1. Notre comité de rédaction est très attaché à sélectionner les articles selon leur qualité, indépendamment du nombre d'articles soumis et avec un nombre maximal de cinq articles par numéro. Or on peut observer que ce nombre fluctue. L'un de nos soucis actuels est de nous assurer d'un nombre stable de soumissions, ce qui devient de plus en plus difficile selon les numéros. Cet aspect est d'autant plus notable pour les deux derniers *Varia* qui peinent à avoir une dizaine de soumissions.

Les statistiques que nous donnons sur l'origine des articles considèrent le pays du premier auteur, hors de France ou pas, ainsi que la langue de la soumission, français en principe ou anglais si l'un des coauteurs n'est pas francophone. Les chiffres sont fournis dans le tableau 2 pour la même période de temps que le tableau 1. En comparant ces chiffres à ceux des derniers numéros, on constate une augmentation récente d'articles acceptés en anglais. Il en va de même pour le nombre de premiers auteurs hors de France, qui après être tombé à 0 durant tout le volume de 2015, remonte progressivement pour arriver à deux articles sur trois exception faite du dernier *Varia*. Cette augmentation est notamment liée à la thématique du numéro 58-3 sur le traitement automatique de l'arabe et des langues apparentées.

Ce numéro contient les articles retenus lors de l'appel non thématique lancé début mars et clos à la mi-juillet 2018. Cet appel portait comme d'habitude sur tous les aspects du traitement automatique des langues. Dix articles ont été soumis dont deux en anglais, ce qui représente un faible nombre de soumissions.

À l'issue du processus de sélection habituel à deux tours, trois articles ont été retenus pour publication. Les tâches abordées par ces articles sont diverses : analyse formelle de la notion de structure syntaxique, étude des algorithmes de la famille « Apprendre à chercher » permettant de réaliser des tâches d'apprentissage structuré, et présentation d'architecture à base de réseaux de neurones pour la désambiguïsation lexicale :

1) « Une approche mathématique de la notion de structure syntaxique : raisonner en termes de connexions plutôt que d'unités », Sylvain Kahane ;

2) « Les méthodes "Apprendre à chercher" en traitement automatique des langues : un état de l'art », Elena Knyazeva, Guillaume Wisniewski et François Yvon ;

3) « Approche supervisée à base de cellules LSTM bidirectionnelles pour la désambiguïsation lexicale », Loïc Vial, Benjamin Lecouteux et Didier Schwab.

On trouvera à la suite des articles des notes de lecture. Nous encourageons nos lecteurs à se faire mutuellement profiter de leurs lectures et à se mettre en contact avec Denis Maurel (`denis.maurel@univ-tours.fr`) pour les publier ici. Suit une liste de résumés de thèses ou d'habilitations à diriger les recherches en traitement automatique des langues préparée par Sylvain Pogodalla. Merci à Denis et Sylvain pour le travail de veille et de collecte.

Enfin, rappelons que la revue TAL reçoit un soutien financier de l'Institut des sciences humaines et sociales du CNRS et de la délégation générale à la langue française et aux langues de France (DGLFLF). Nous adressons nos remerciements à ces organismes.

Emmanuel Morin
LS2N, Université de Nantes
emmanuel.morin@univ-nantes.fr

Sophie Rosset
LIMSI, CNRS
sophie.rosset@limsi.fr

Pascale Sébillot
IRISA, INSA Rennes
pascale.sebillot@irisa.fr

Merci aux relecteurs spécifiques de ce numéro :

Richard Moot, LIRMM, CNRS
Christian Raymond, IRISA, INSA Rennes
Tim Van De Cruys, IRIT, CNRS

ainsi qu'aux membres du comité de rédaction de la revue (voir sa composition sur notre site).

Une approche mathématique de la notion de structure syntaxique : raisonner en termes de connexions plutôt que d'unités

Sylvain Kahane

Modyco, Université Paris Nanterre et CNRS

sylvain@kahane.fr

RÉSUMÉ. *Cet article propose d'évaluer l'équivalence des structures syntaxiques (arbres de constituants, arbres de dépendance, etc.) en fonction des combinaisons d'unités qu'elles définissent et des connexions que cela induit. La notion de connexion est définie comme une classe d'équivalence de combinaisons, ce qui nous permet d'introduire une notion de structure syntaxique qui s'abstrait en partie de la notion d'unité syntaxique. Les conséquences que cela a sur la conception de la structure syntaxique sont présentées.*

ABSTRACT. *A mathematical approach of the notion of syntactic structure: reasoning in terms of connections rather than units.*

The aim of this paper is to evaluate syntactic structures (constituency trees, dependency trees. etc.) according to the combinations of units that they define and the connections that are induced. The notion of connection is defined as an equivalence class of combinations, which allows us to introduce a notion of syntactic structure that cut itself off from the notion of syntactic unit. Consequences of that on the conception of the syntactic structure are investigated

MOTS-CLÉS : *connexion syntaxique, combinaison syntaxique, dépendance, constituance, relation d'équivalence, graphe à bulles.*

KEYWORDS: *syntactic connection, syntactic combination, dependency, constituency, equivalence relation, bubble graph.*

Préambule. Cet article n'est pas à proprement parler du TAL. Il s'agit de linguistique mathématique. Néanmoins, nous pensons que le public auquel cet article s'adresse est en partie celui de la revue TAL et que les résultats théoriques que nous présentons ici peuvent avoir des incidences sur la modélisation des langues et leur implémentation.

1. Introduction

L'objectif de cet article est de s'interroger sur la nature formelle de la structure syntaxique. Il ne s'agit pas de discuter les critères linguistiques qui permettent de définir la structure syntaxique ou de montrer quelle structure permet de modéliser au mieux tel ou tel phénomène syntaxique en fonction des critères retenus. Nous nous intéressons ici à la nature mathématique de la structure en fonction des objets linguistiques considérés par la théorie. Nous pensons que la conception que les linguistes ont de la structure syntaxique et des objets qu'encode cette structure (constituants ou dépendances par exemple) est directement contrainte par les objets formels que nous utilisons pour les représenter. Les modes de représentation que nous utilisons, comme les arbres de constituants ou les arbres de dépendance, limitent notre conception de la structure syntaxique et peuvent induire des conceptions erronées de la nature de cette structure. Une meilleure compréhension de la structure syntaxique passe, à notre avis, par l'introduction de structures mathématiques plus riches que les traditionnels arbres, les arbres en question n'étant que des vues particulières et réduites de la véritable structure.

Nous souhaitons en particulier mettre au centre de la discussion la notion de *connexion*, qui est, à notre avis, une notion naturelle, mais formellement complexe, puisqu'une connexion est un ensemble de combinaisons équivalentes, ensemble que l'on peut difficilement représenter par un diagramme de type graphe. Les diagrammes habituellement utilisés en linguistique formelle, et notamment les arbres de constituants et de dépendance, ne sont de ce point de vue que des représentants particuliers des structures abstraites que nous allons introduire. De telles vues, si elles restent utiles pour le linguiste, doivent être appréhendées comme des vues simplifiées de la nature réelle de la structure. Nous pensons que prendre en compte la structure dans toute sa complexité formelle permet de mieux comprendre pourquoi les linguistes proposent des vues si variées de la structure sans arriver à un accord. Nous pensons également que prendre en compte toute la richesse de la structure permettrait d'envisager des traitements automatiques plus élégants, cependant cette dernière question ne sera qu'effleurée dans cet article qui se limite à présenter des objets mathématiques pour définir la structure.

Dans la section 2, nous présenterons quelques exemples de la diversité des diagrammes syntaxiques considérés dans la littérature. Ce n'est pas la diversité des analyses linguistiques qui nous intéresse ici, mais bien la diversité des objets mathématiques considérés. Dans la section 3, nous montrons comment passer d'une évaluation de la structure en termes d'unités à une évaluation en termes de

combinaisons, puis de connexions. La section 4 introduit la notion formelle de combinaison et montre l'interprétation de différentes structures syntaxiques en termes d'ensembles de combinaisons. La section 5 rappelle les notions de relations d'équivalence et de relation d'ordre dont nous ferons grand usage dans la suite de l'article. Dans la section 6, nous introduisons une relation de compatibilité entre combinaisons, qui nous permet de définir les notions formelles de connexion et d'ensemble de connexions. La section 7 montre comment associer à un arbre de dépendance un ensemble de combinaisons plus riche, puis étend les notions d'ensemble de combinaisons minimal et maximal à tout type de structure. Nous étudions dans la section 8 différentes relations de compatibilité, d'ordre ou d'équivalence entre les ensembles de combinaisons, ce qui nous permet de donner une cartographie des différents ensembles de connexions envisageables. La section 9 revient sur la question des unités à travers la notion de granularité. Nous concluons à la section 10, en tirant les conséquences de notre étude en ce qui concerne la notion de structure syntaxique et le statut des diagrammes syntaxiques les plus utilisés, les arbres de constituants et les arbres de dépendance.

2. Diversité mathématique des structures syntaxiques

Rappelons que nous appelons *arbre* un graphe orienté dont les arêtes, appelées également branches, vont d'un nœud appelé le *gouverneur* à un nœud appelé le *dépendant*. Pour être un arbre, ce graphe doit être connexe et acyclique et chaque nœud doit posséder un gouverneur à l'exception d'un nœud qu'on appelle la racine de l'arbre. Dans cet article, nous ne nous intéresserons pas à l'ordre linéaire et nos arbres n'incluent pas d'ordre additionnel sur les nœuds.

Commençons par les *arbres de constituants*, comme l'arbre T0 de la figure 1 qui représente une analyse en constituants immédiats de la phrase (1). Cet arbre indique que *le* et *chat* forment ensemble une unité *le chat* ou, de manière équivalente, que *le chat* a pour constituants immédiats *le* et *chat*. De même, *vu* et *Zoé* forment ensemble une unité *vu Zoé* qui se combine avec *a* pour former une unité *a vu Zoé* qui à son tour se combine avec *le chat* pour donner la phrase entière. Il est intéressant de noter que dans ses premiers travaux, Noam Chomsky ne représente pas l'analyse en constituants immédiats par un arbre, mais par une structure un peu différente, comme on peut le voir sur le diagramme de droite la figure 1 extrait de (Chomsky, 1957). Plus précisément, la règle Sentence → NP + VP est représentée par un lien entre NP et VP indiquant que ces deux constituants se combinent entre eux, puis par un petit lien entre ce premier lien et Sentence indiquant que le résultat de la combinaison donne Sentence. La structure n'est donc pas un graphe[1], mais un

1 Comme l'ont montré Mazziotta et Kahane (2017), il ne semble pas y avoir eu de représentation des analyses en constituants immédiats sous forme d'arbre (c'est-à-dire de graphe pointé connexe et acyclique) avant Chomsky (1956) et toutes les représentations proposées avant cela mettent en avant la connexion. Le diagramme proposé par Chomsky (1957), réifiant les connexions, s'inscrit donc parfaitement dans la tradition de son époque.

polygraphe (Kahane et Mazziotta, 2015), où une arête a pour sommet une autre arête. On notera qu'une telle représentation, qui encode explicitement les connexions (comme la connexion entre NP et VP), n'est possible que pour des grammaires qui n'ont que des décompositions binaires[2].

(1) *Le chat a vu Zoé.*

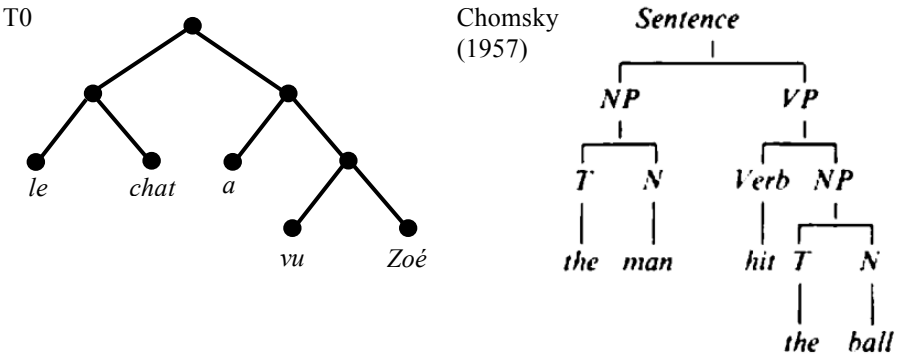

Figure 1. Arbre de constituants binaire (T0) et diagramme de Chomsky (1957)

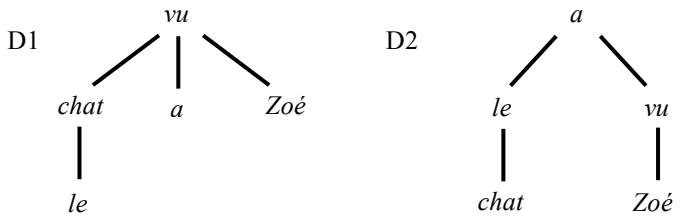

Figure 2. Arbres de dépendance à la UD (D1) et à la Hudson (D2)

Passons aux *arbres de dépendance*. La figure 2 propose deux arbres de dépendance pour la phrase (1) : D1 est un arbre selon le schéma Universal Dependencies [UD] (Nivre *et al.*, 2016), où les liens entre les mots pleins sont

2 Dans cet article, nous ne nous intéresserons qu'aux arbres de constituants binaires. D'une certaine façon, le formalisme des arbres de constituants est utilisé de deux façons assez différentes dans la littérature. Les arbres de constituants binaires cherchent à représenter des combinaisons binaires d'unités : autrement dit, les deux constituants immédiats d'une unité donnée se combinent entre eux pour former cette unité. L'interprétation des arbres plats à branchements multiples est différente. Dans une phrase comme *Pierre donne un livre à Marie*, si l'on considère VP → V NP PP, on ne veut pas dire que NP et PP se combinent entre eux, mais bien que c'est V qui se combine avec NP et avec PP. La règle VP → V NP PP n'est donc réellement interprétable que si l'on ajoute que V est la tête de VP et l'on voit alors que ce branchement encode en fait deux combinaisons, V avec NP et V (ou V NP) avec PP.

privilégiés, D2 un arbre à la Hudson (1984 ; 2007), où les mots fonctionnels sont traités comme des têtes. Si les analyses linguistiques sont différentes et obéissent à des critères différents, le formalisme est exactement le même, chaque branche représentant la combinaison entre deux mots.

On peut en fait donner d'autres interprétations des branches d'un arbre de dépendance et voir la combinaison non pas entre deux mots, mais entre un mot et une unité qui serait la projection du nœud dépendant. La *projection* d'un nœud *x* est l'unité formée des nœuds dominés par *x*, *x* inclus. Ainsi, la dépendance entre *a* et *vu* de D2 peut être interprétée comme une combinaison entre *a* et la projection de *vu*, c'est-à-dire *vu Zoé*. Cette double interprétation de l'analyse en dépendance a déjà été mise en évidence par Nicolas Beauzée (1765) dans l'article *Régime* de l'*Encyclopédie* de Diderot et d'Alembert. Elle a été formalisée par Gladkij (1968) qui propose une structure de type arbre à bulles (Kahane, 1997), où les branches peuvent lier des bulles qui possèdent elles-mêmes une analyse interne. La structure S2 de la figure 3 est un « arbre » à la Beauzée-Gladkij pour la phrase (1) suivant la même analyse qu'en D2.

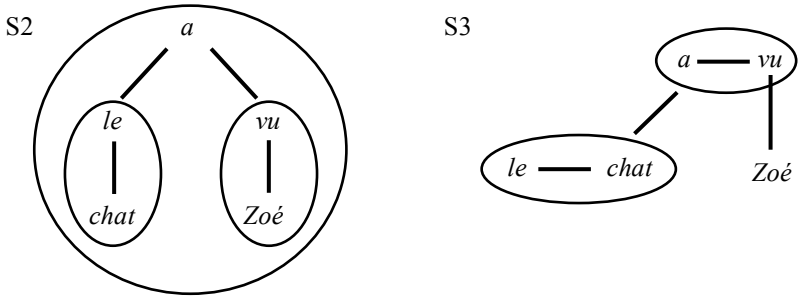

Figure 3. Un « arbre » à la Beauzée-Gladkij et un stemma à la Tesnière

Considérons, pour terminer, les structures introduites par Lucien Tesnière (1959). Si Tesnière est généralement crédité d'avoir marqué un pas décisif dans le développement de la syntaxe de dépendance, force est de constater que ses stemmas ne sont généralement pas des arbres. L'introduction des nucléus est particulièrement intéressante. Tesnière considère que les mots grammaticaux comme les auxiliaires ou les articles forment avec les mots pleins des nucléus et que c'est entre les nucléus que les connexions ont lieu. La structure S3 de la figure 3 donne un stemma à la Tesnière pour la phrase (1). Dans ce stemma, *a* et *vu* se combinent entre eux et forment le nucléus *a vu* dont dépend le nucléus *le chat*, qui résulte de la combinaison de *le* et *chat*, tandis que *Zoé* dépend de *vu* seul et non du nucléus *a vu*.[3]

3 Selon les stemmas, Tesnière utilise, pour représenter les nucléus (translatifs), une notation en T ou une bulle comme ici. Dans les notations en T, l'élément lexical peut avoir ses propres dépendants. C'est l'analyse que nous avons retenue ici en faisant dépendre *Zoé* du verbe lexical *vu* et non de la totalité du nucléus. Nous avons par ailleurs ajouté un lien entre les deux

3. Des unités aux connexions

La question des unités syntaxiques est devenue centrale en linguistique à partir de l'émergence de l'analyse en constituants immédiats (Bloomfield, 1933). Définir la structure consiste alors à déterminer les constituants majeurs de la phrase, puis les constituants immédiats de chaque constituant et ainsi de suite. L'amorce même du processus nécessite de déterminer l'unité maximale de la syntaxe, la phrase. On retrouve cette contrainte jusqu'à aujourd'hui en TAL, puisque la quasi-totalité des analyseurs syntaxiques automatiques suppose une segmentation préalable du texte à analyser en phrases.

Pour comparer deux analyses en constituants, on compare généralement les unités considérées. Il en va de même pour comparer une analyse en dépendance et une analyse en constituants : la méthode la plus courante consiste à regarder la projection de chacun des éléments de l'arbre de dépendance et de les comparer avec les constituants de l'analyse en constituants (Lecerf, 1961 ; Kahane, 2001).

Nous pensons que ce n'est pas en termes d'unités, mais plutôt en termes de combinaisons que les analyses syntaxiques doivent être comparées. Expliquons-nous sur un exemple :

(2) *Le petit frère de Zoé m'a parlé de ça*.

Une analyse en constituants immédiats (Bloomfield, 1933 ; Chomsky, 1957) va postuler que la phrase (S) est la combinaison d'un NP (*le petit frère de Zoé*) et d'un VP (*m'a parlé de ça*), une analyse traditionnelle que la forme verbale *a parlé* a pour sujet *le petit frère de Zoé*, une analyse en dépendance à la UD qu'il y a une dépendance entre *parlé* et *frère* (UD), d'autres analyses encore pourront considérer que la dépendance est entre l'auxiliaire *a* et le déterminant *le* (Hudson, 1984 ; 2007) ou entre les nucléus *a parlé* et *le frère* (Tesnière, 1959) ou encore entre le chunk verbal *m'a parlé* et le chunk nominal *le petit frère* (Vergne, 2000). D'autres encore peuvent considérer que c'est en fait la flexion verbale ou le mode indicatif qui gouverne le sujet (ce que sous-entend la décomposition IP → DP + I' (Abney, 1987)). Mais finalement toutes ces analyses s'accordent sur le fait qu'il y a, à un certain endroit, une *connexion*, que nous appellerons la *connexion subjectale*, entre une unité verbale et son sujet, et que cette connexion est instanciée de différentes façons selon les théories et les formalismes sous-jacents (figure 4). Certaines de ces analyses sont compatibles, d'autres sont concurrentes, certaines analyses sont plus fines que d'autres. Nous allons donner un sens mathématique à ce que nous venons de dire et voir ce que l'on peut en faire.

éléments du nucléus pour bien indiquer qu'ils sont combinés. Voir également Kahane et Osborne (2015) pour l'interprétation des stemmas et une représentation en termes de polygraphe.

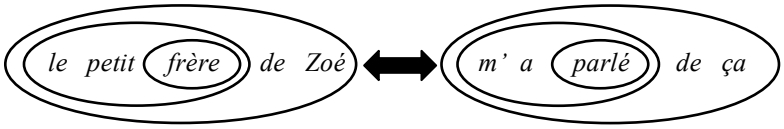

Figure 4. Différentes instanciations de la connexion subjectale

4. Combinaisons et ensembles de combinaisons

Même si, comme nous le verrons, on peut s'abstraire en partie de la question des unités, notre définition des combinaisons repose principalement sur la notion d'unité. Nous appelons *unité* tout signe linguistique, vu ici comme une portion d'un texte. Notre définition est très lâche : nous ne faisons pas, en particulier, d'hypothèse de continuité. Les unités sont des suites (éventuellement discontinues) d'unités élémentaires, que nous traiterons comme des ensembles d'unités élémentaires (en particulier, nous considérerons l'intersection ou la réunion de deux unités). Dans cet article, nous considérons que les unités élémentaires d'une phrase sont les mots[4], mais la méthode peut, bien sûr, s'appliquer à des décompositions morphosyntaxiques plus fines en lexèmes et morphèmes flexionnels. Nous reviendrons sur la question de la granularité de la structure dans la section 9.

L'analyse d'une phrase, quel que soit le formalisme, repose au départ sur l'identification d'unités et de combinaisons entre ces unités. Nous nous limiterons dans cet exposé aux combinaisons binaires même si l'on peut tout à fait envisager d'étendre ce travail à des combinaisons ternaires (par exemple avec le traitement de certains morphèmes comme des marqueurs de connexions)[5]. Il y a deux principales façons d'indiquer que deux unités se combinent : soit en indiquant que ces deux unités forment ensemble une autre unité (c'est la méthode de l'analyse en constituants immédiats), soit en indiquant explicitement qu'il y a une connexion entre les deux unités (c'est la méthode des grammaires de dépendance). Il a déjà été montré que les deux méthodes sont en grande partie équivalentes (voir la notion de *grouping* chez Kahane et Mazziotta, 2015), mais toutes les conséquences n'en ont pas été tirées. Nous représentons une combinaison par une paire (non ordonnée)[6] de

4 Il s'agit plus précisément des occurrences de mots (les différentes occurrences d'un même mot doivent être distinguées).

5 La considération de dépendances ternaires apparaît à plusieurs reprises dans l'histoire de la syntaxe de dépendance. On en trouve chez Tesnière (1934) pour la coordination (la conjonction de coordination *et* étiquette la relation entre les conjoints), chez Débili (1982) pour les prépositions régimes et plus récemment dans les Collapsed Stanford Dependencies (CSD) (de Marneffe et Manning, 2008).

6 On pourrait prendre en compte la notion de *tête* en travaillant avec des couples (A,B) et la convention que le premier élément est la tête de la combinaison. Tout ce que nous faisons dans cet article peut être étendu à cette situation. Considérer à la fois des combinaisons avec

deux unités, c'est-à-dire un ensemble à deux éléments. Pour que {A,B} soit une *combinaison*, il faut que A et B soient des unités non vides et que A et B soient disjoints ($A \cap B = \varnothing$).

Un arbre de constituants binaire T induit naturellement un ensemble de combinaisons $C(T)$: à chaque fois qu'un constituant C se décompose en deux constituants A et B (ce que l'on note habituellement par $C \rightarrow A\ B$), on postule une combinaison {A,B}, puisque A et B se combinent pour donner C (cf. Chomsky, 1957 ; Nida, 1966 ; Mazziotta et Kahane, 2017, sur l'interprétation de la constituance en termes de connexions). Si nous reprenons l'arbre de constituants T0 de la section 2, nous avons :

$$C(T0) = \{\ \{le\ chat,\ a\ vu\ Zoé\},\ \{le,\ chat\},\ \{a,\ vu\ Zoé\},\ \{vu,\ Zoé\}\ \}$$

On peut associer à tout ensemble de combinaisons une représentation, que nous appellerons un *graphe à bulles*, où les unités non élémentaires sont représentées par des bulles et les combinaisons par des arêtes entre les unités. On associe ainsi à $C(T0)$ le graphe à bulles B0 de la figure 5.

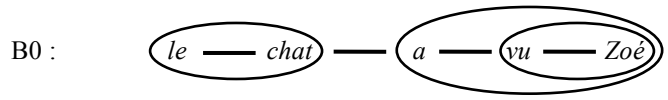

Figure 5. Graphe à bulles associé à $C(T0)$

Un arbre de dépendance D induit également naturellement un ensemble de combinaisons $C_{min}(D)$: si deux unités A et B sont connectées dans l'arbre de dépendance (A et B sont généralement des mots), alors on postule une combinaison {A,B}. Si nous reprenons l'arbre de dépendance D1 de la section 2, nous avons :

$$C_{min}(D1) = \{\ \{le,\ chat\},\ \{chat,\ vu\},\ \{a,\ vu\},\ \{vu,\ Zoé\}\ \}$$

Avec les mêmes conventions que précédemment, on peut associer un graphe à bulles B1 à $C_{min}(D1)$ (figure 6). B1, qui ne contient que des unités élémentaires et donc pas de bulles à proprement parler, est le graphe (non orienté) sous-jacent à l'arbre D1 :

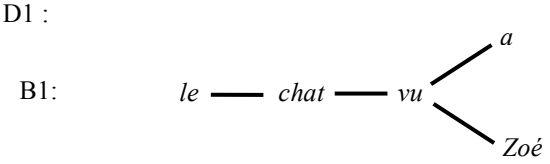

Figure 6. Graphe à bulles associé à $C_{min}(D1)$

Plus généralement, à tout diagramme comportant l'identification d'unités (par exemple, par des bulles) et de liens binaires entre les unités, on peut associer l'ensemble de combinaisons formé de toutes les paires d'unités liées par un lien

---

et sans tête demanderait davantage de précautions. Ce problème est comparable à la prise en compte à la fois d'arêtes orientées et non orientées dans un graphe.

binaire. Les liens binaires entre les unités sont généralement représentés soit par des arêtes entre les unités (comme dans le cas des arbres de dépendance ou de structures plus complexes telles que S2 ou S3), soit par l'enchâssement d'unités (comme dans les arbres de constituants). On peut ainsi associer des ensembles de combinaisons aux structures S2 et S3 de la figure 3 :

C_{min}(S2) = { {*le, chat*}, {*vu, Zoé*}, {*le chat, a*}, {*a, vu Zoé*} }

C_{min}(S3) = { {*le, chat*}, {*vu, Zoé*}, {*a, vu*}, {*le chat, a vu*} }

Les représentations associées à ces ensembles de combinaisons, donnés figure 7, sont les mêmes que ceux de la figure 3, à la différence que la hiérarchie induite par le positionnement vertical n'est plus prise en compte.

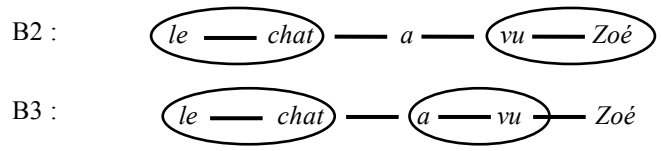

Figure 7. Graphes à bulles associés à C_{min}(S2) et C_{min}(S3)

5. Notions mathématiques

La suite de cet article fait appel à des notions de théorie des ensembles qu'il nous semble préférable de rappeler. Si E est un ensemble, on note E×E l'ensemble des couples d'éléments de E. Tout sous-ensemble de E×E peut être vu comme une *relation binaire* sur les éléments de E. On dit d'une relation binaire R qu'elle est *réflexive* si tout élément de E est en relation avec lui-même ($\forall x$, xRx), *symétrique* si toute relation peut être inversée ($\forall x,y$, xR$y \rightarrow y$Rx), *antisymétrique* si au contraire elle ne peut jamais être inversée sauf dans les cas triviaux ($\forall x,y$ [xRy et yRx] $\rightarrow$ $x = y$) et *transitive* si elle se propage systématiquement ($\forall x,y,z$ [xRy et yRz] $\rightarrow$ xRz).

Une relation qui est réflexive, symétrique et transitive est appelée une *relation d'équivalence*. Une relation d'équivalence définit une partition de E, c'est-à-dire qu'elle découpe E en sous-ensembles deux à deux disjoints au sein desquels tous les éléments sont en relation les uns avec les autres. Ces ensembles sont appelés les *classes d'équivalence* de la relation R. L'ensemble des classes d'équivalence de la relation R sur E est appelé *ensemble quotient* de E par R et est noté $E_{/R}$. Il paraît un peu difficile de se représenter un tel ensemble au premier abord, car il s'agit d'un ensemble d'ensembles. Si on note $\mathcal{P}$(E) l'*ensemble des parties* de E (c'est-à-dire les ensembles inclus dans E), les éléments de $E_{/R}$ appartiennent à $\mathcal{P}$(E) et donc $E_{/R}$ est une partie de $\mathcal{P}$(E), c'est-à-dire un élément de $\mathcal{P}(\mathcal{P}(E))$. Nous allons beaucoup utiliser la notion d'ensemble quotient et travailler non seulement avec des ensembles d'ensembles, mais avec des ensembles d'ensembles d'ensembles et ainsi de suite. Nous noterons $\mathcal{P}^2(E) = \mathcal{P}(\mathcal{P}(E))$, $\mathcal{P}^3(E) = \mathcal{P}(\mathcal{P}^2(E))$ et ainsi de suite.

Nous allons manipuler deux notions qu'il nous faudra bien distinguer : les combinaisons qui sont des paires d'unités, comme on vient de le voir, et les connexions qui sont des ensembles de combinaisons équivalentes, pour une relation d'équivalence que nous allons définir dans la prochaine section. On part d'un ensemble X d'unités élémentaires, les mots d'une phrase dans nos exemples. On considère l'ensemble U des unités sur X, qui sont des parties de X, que nous représentons par des suites sur X par souci de lisibilité, bien que l'ordre linéaire ne soit pas pris en compte dans cette étude. Ainsi la suite *a vu Zoé* représente en fait l'ensemble $\{a, vu, Zoé\}$. Les combinaisons sont des paires d'unités et donc des éléments de $\mathcal{P}(U)$. Les connexions sont des éléments de $\mathcal{P}^2(U)$, c'est-à-dire de $\mathcal{P}^3(X)$.

Pour ceux qui ne sont pas familiers avec les ensembles quotients, il est certainement utile de faire le parallèle avec l'ensemble des nombres rationnels. On sait que $1/2$ ou $2/4$ ou encore $50/100$ représentent le même nombre rationnel. Autrement dit, l'ensemble des rationnels est un ensemble quotient obtenu à partir de l'ensemble des couples de nombres entiers et de la relation d'équivalence suivante : deux couples (a,b) et (a',b') représentent le même nombre rationnel si et seulement si $ab' = a'b$. Chaque couple (a,b), ou fraction a/b pour prendre la notation usuelle, est un représentant du nombre relationnel. De même, les combinaisons sont des *représentants* des connexions.

Revenons sur les relations générales. À partir de n'importe quelle relation R, on peut définir une relation R* qu'on appelle la *clôture réflexive et transitive* de R. L'idée est que deux éléments x et y sont en relation par R* (xR*y) s'il existe une chaîne d'éléments en relation par R qui va de x à y, ce qui se formalise par :

$$x\mathrm{R}^*y \text{ ssi } \exists n \geq 0,\ \exists x_0, x_1, \ldots, x_n \text{ tels que } x = x_0, y = \mathrm{x_n} \text{ et } \forall i = 0,\ldots,n\text{-}1\ x_i\mathrm{R}x_{i+1}$$

Par exemple, si R est la *relation de dépendance* dans un arbre (xRy ssi y dépend de x), la relation R* est la *relation de dominance* (xR*y ssi x domine y). La projection de x (voir section 2) est l'unité formée par l'ensemble des nœuds dominés par x.

Nous allons maintenant considérer un autre type de relation, tout aussi utile que les relations d'équivalence. Une relation qui est réflexive, antisymétrique et transitive est appelée un *ordre* (ang. *partial order*). Par exemple, la relation $\leq$ est un ordre sur les nombres entiers. Un ordre qui s'apparente davantage aux ordres que nous considérons dans cet article est la relation d'inclusion $\subseteq$ entre ensembles. Cet ordre est *partiel* sur $\mathcal{P}(E)$ dans le sens où de nombreux couples de parties de E ne sont pas ordonnés l'un par rapport à l'autre, comme par exemple les paires $\{a,b\}$ et $\{b,c\}$. Un ordre partiel de ce type peut être représenté par un treillis. Nous donnons figure 8 le treillis de l'inclusion sur $\mathcal{P}(\{a,b,c\})$. En fait, ce treillis ne représente pas directement la relation d'inclusion, mais une relation de « précédence » immédiate R dont la relation d'inclusion est la clôture réflexive et transitive ($\subseteq$ = R*).

Pour un ordre (partiel ou non), on peut définir plusieurs notions. On appelle *maximum* un élément qui est plus grand que tous les autres et *minimum* un élément

qui est plus petit que tous les autres. E est le maximum de l'inclusion sur $\mathcal{P}$(E), tandis que l'ensemble vide, noté ∅, est le minimum. On appelle *supremum* de *x* et *y*, noté $x \vee y$, le plus petit des éléments plus grands que *x* et *y* et *infimum* de *x* et *y*, noté $x \wedge y$, le plus grand des éléments plus petits que *x* et *y*. Les supremums et infimums pour l'inclusion sont respectivement l'union et l'intersection : $A \vee B = A \cup B$; $A \wedge B = A \cap B$. On repère facilement les supremums et infimums de *x* et *y* dans le treillis correspondant à un ordre, puisqu'il s'agit des éléments directement liés à *x* et *y* qui se trouvent respectivement au-dessus ou en dessous.

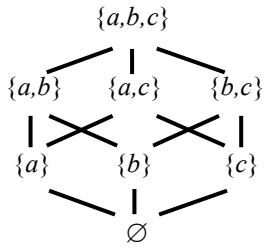

Figure 8. Treillis pour l'inclusion sur $\mathcal{P}(\{a,b,c\})$

Une relation qui est seulement réflexive et transitive est un *préordre* (Bourbaki, 1939). Si la relation $\preccurlyeq$ est un préordre, elle définit naturellement une relation d'équivalence ≡ par la relation suivante : $x \equiv y$ si et seulement si $x \preccurlyeq y$ et $y \preccurlyeq x$. On peut alors projeter la relation $\preccurlyeq$ sur l'ensemble quotient $E_{|\equiv}$ en considérant que deux classes d'équivalence X et Y vérifient $X \preccurlyeq Y$ si pour tous éléments *x* de X et *y* de Y, on a $x \preccurlyeq y$. La relation $\preccurlyeq$ est un ordre sur $E_{|\equiv}$.

6. Compatibilité de combinaisons et connexions

On peut maintenant introduire une relation binaire sur les combinaisons que nous notons ≈ et que nous appelons la relation de *compatibilité* :

$\{A,B\} \approx \{A',B'\}$ ssi $A \cap A'$ et $B \cap B'$ sont non vides
et $A \cup A'$ et $B \cup B'$ sont disjoints.[7]

La première condition ($A \cap A'$ et $B \cap B'$ non vides) assure que $\{A \cap A', B \cap B'\}$ est encore une combinaison. La deuxième condition ($A \cup A'$ et $B \cup B'$ disjoints) assure que $\{A \cup A', B \cup B'\}$ est également une combinaison. Autrement dit :

$\{A,B\} \approx \{A',B'\}$ ssi $\{A \cap A', B \cap B'\}$ et $\{A \cup A', B \cup B'\}$
sont également des combinaisons.

Dire que deux combinaisons sont compatibles revient à dire qu'elles peuvent représenter la même connexion, en l'instanciant chacune à leur façon. $\{A \cap A',$

7 Rappelons que, par définition de la notion de combinaison, on a aussi A et B disjoints, ainsi que A' et B'.

B ∩ B'} et {A ∪ A', B ∪ B'} seront également des instanciations possibles de cette connexion. On a en fait :

{A,B} ≈ {A',B'} ≈ {A ∩ A', B ∩ B'} ≈ {A ∪ A', B ∪ B'}.

On peut représenter la relation de compatibilité par la configuration de la figure 9.

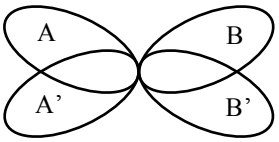

Figure 9. Deux combinaisons {A,B} et {A',B'} compatibles

La relation ≈ est réflexive et symétrique, mais elle n'est pas en général transitive. Si nous considérons les connexions subjectales de T0, D1 et D2, on récupère les combinaisons respectives suivantes :

c0 = {*le chat, a vu Zoé*}, c1 = {*chat, vu*} et c2 = {*le, a*}.

On a c1 ≈ c0 et c0 ≈ c2, mais c1 $\not\approx$ c2, ce qui montre la non-transitivité.

On peut éviter la non-transitivité en ne prenant en compte qu'une partie des combinaisons. La relation ≈ est une relation d'équivalence sur de nombreux sous-ensembles $\mathcal{E}$ de combinaisons. L'avantage d'une relation d'équivalence est qu'elle induit une partition de l'ensemble (section 5). Dans le cas d'un ensemble de combinaisons $\mathcal{E}$, les éléments de l'espace quotient $\mathcal{E}_{/\approx}$ sont appelés des *connexions*. Autrement dit, à partir de nos structures de départ, nous induisons des ensembles de combinaisons, puis des ensembles de connexions.

Les *ensembles de connexions*, du type $\mathcal{E}_{/\approx}$ où $\mathcal{E}$ est un ensemble de combinaisons, sont, selon notre point de vue, les véritables structures syntaxiques. Avant d'expliquer pourquoi, nous allons présenter différents exemples d'ensembles de connexions, ainsi que différentes relations entre ces ensembles, ce qui nous permettra d'avoir une bonne idée des différentes structures possibles et de la place qu'ont les arbres de constituants et les arbres de dépendance dans ce paysage.

Reprenons les exemples introduits dans la section 4 et considérons maintenant l'ensemble de combinaisons $\mathcal{E}4 = C_{\min}(D1) \cup C(T0)$ = { {*le, chat*}, {*chat, vu*}, {*le chat, a vu Zoé*}, {*a, vu*}, {*a, vu Zoé*}, {*vu, Zoé*} }. La relation ≈ est bien une relation d'équivalence sur $\mathcal{E}4$. Les classes d'équivalence sont :

C0 = { {*le, chat*} }
C1 = { {*chat, vu*}, {*le chat, a vu Zoé*} }
C2 = { {*a, vu*}, {*a, vu Zoé*} }
C3 = { {*vu, Zoé*} }

Chacune de ces classes est une connexion et $\mathcal{E}4_{/\approx}$ = { C0, C1, C2, C3 } est un ensemble de connexions. Rappelons que si U est l'ensemble des unités, les combinaisons sont des éléments de $\mathcal{P}$(U), les ensembles de combinaisons, comme C_{min}(D1), C(T0) et $\mathcal{E}4$, et les connexions sont des éléments de $\mathcal{P}^2$(U) et les ensembles de connexions, comme $\mathcal{E}4_{/\approx}$, sont eux des éléments de $\mathcal{P}^3$(U).

Les connexions C1 et C2 ont deux représentants. L'un des représentants appartient à C_{min}(D1) et l'autre à C(T0). On peut voir les ensembles de combinaisons C_{min}(D1) et C(T0) comme des instanciations particulières de l'ensemble de connexions $\mathcal{E}4_{/\approx}$, où une combinaison a été sélectionnée dans chaque connexion. Nous allons maintenant préciser le lien entre les différents ensembles de connexions et leurs différentes instanciations, les ensembles de combinaisons.

7. Ensembles de combinaisons minimal et maximal

On peut associer à un arbre de dépendance D un ensemble de combinaisons *étendu* que nous appelons E_{max}(D) et qui est, d'un certain point de vue que nous allons définir, équivalent à E_{min}(D). Nous considérons maintenant comme unités toutes les portions connexes de notre arbre de dépendance, encore appelées les *catenae* par (Osborne *et al.*, 2012). Les catenae de D1 sont :

Catenae(D1) = { *le, chat, a, vu, Zoé, le chat, chat vu, a vu, vu Zoé, le chat vu, chat a vu, chat vu Zoé, a vu Zoé, le chat a vu, le chat vu Zoé, chat a vu Zoé, le chat a vu Zoé* }.

À la suite de Gerdes et Kahane (2013), on peut, à partir de tout ensemble X d'unités, définir un ensemble de combinaisons, que nous appelons Combi(X) :

{A,B} appartient à Combi(X) ssi A, B et A ∪ B sont des éléments de X.

Nous pouvons ainsi définir, à partir d'un arbre de dépendance D, un nouvel ensemble de combinaisons, que nous nommons C_{max}(D) :

C_{max}(D) = Combi(Catenae(D)).

On peut construire C_{max}(D) directement à partir de D : deux portions connexes A et B de D forment une combinaison {A,B} de C_{max}(D) ssi A et B sont disjointes et connectées par une dépendance. L'ensemble de combinaisons C_{max}(D) obtenu est assez gros. Ainsi C_{max}(D1) contient à la fois C_{min}(D1) et C(T0) et encore plein d'autres combinaisons. Malgré cela, ≈ est une relation d'équivalence sur cet ensemble. En fait, les C_{max}(D) sont les plus gros ensembles sur lesquels ≈ est une relation d'équivalence.

Nous allons maintenant nous intéresser aux ensembles de connexions $C_{max}(D)_{/\approx}$. Tout ensemble $C_{max}(D)_{/\approx}$ contient le même nombre de connexions que $C_{min}(D)_{/\approx}$. Par exemple, les connexions dans $C_{max}(D1)_{/\approx}$ sont :

C'0 = { {*le, chat*}, {*le, chat vu*}, {*le, chat a vu*}, {*le, chat vu Zoé*}, {*le, chat a vu Zoé*} }

C'1 = { {*chat, vu*}, {*le chat, vu*}, {*chat, a vu*}, {*chat, vu Zoé*}, {*le chat, a vu*}, {*le chat, vu Zoé*}, {*chat, a vu Zoé*}, {*le chat, a vu Zoé*} }
C'2 = { {*a, vu*}, {*a, vu Zoé*}, {*a, chat vu*}, {*a, le chat vu*}, {*a, chat vu Zoé*}, {*a, le chat vu Zoé*} }
C'3 = { {*vu, Zoé*}, {*a vu, Zoé*}, {*chat vu, Zoé*}, {*le chat vu, Zoé*}, {*le chat a vu, Zoé*} }

On voit que chaque connexion dans l'arbre de dépendance correspond à tout un ensemble de combinaisons et pas seulement à une combinaison entre mots. On peut représenter cet ensemble de combinaisons équivalentes par un diagramme, comme nous le proposons dans la figure 10 pour la connexion subjectale C'1 de $C_{max}(D1)_{/\approx}$.

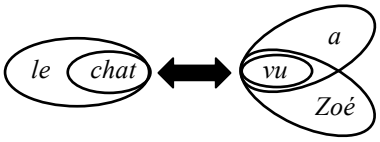

Figure 10. Les combinaisons de la connexion subjectale C'1 de $C_{max}(D1)/\approx$

Nous avons associé deux ensembles de combinaisons à tout arbre de dépendance, $C_{min}(D)$ et $C_{max}(D)$. Si nous avons appelé ces deux ensembles C_{min} et C_{max}, c'est qu'ils sont, d'un certain point de vue (qu'il nous reste à définir), le plus petit et le plus grand ensemble de combinaisons qu'on peut associer à un arbre de dépendance.

$C_{min}(D)$ est clairement le plus petit ensemble de combinaisons qu'on peut associer à D, car il ne contient qu'une combinaison par connexion, c'est-à-dire par classe d'équivalence de la relation ≈. Tout ensemble de combinaisons qui ne contiendrait qu'une combinaison par connexion et qui serait différent d'un $C_{min}(D)$ représente une autre structure, « autre » en un sens qu'il nous reste à définir. $C_{max}(D)$ est le plus grand ensemble de combinaisons qu'on peut associer à D, sous certaines conditions de connexité et d'acyclicité de l'ensemble de combinaisons que nous allons introduire plus bas.

Ces deux ensembles nous intéressent pour des raisons opposées : l'avantage d'un ensemble comme $C_{min}(D)$, avec un seul représentant par connexion, est qu'il peut facilement donner lieu à une représentation graphique, un *diagramme syntaxique* comme ceux des figures 5, 6 et 7. Mais fondamentalement, un tel diagramme doit être interprété comme $C_{max}(D)$, c'est-à-dire comme contenant un foisonnement de combinaisons possibles.

Nous voudrions généraliser ce que nous avons fait pour les ensembles de combinaisons associés aux arbres de dépendance à tous les ensembles de combinaisons. Autrement dit, nous voulons associer à tout ensemble de combinaisons $\mathcal{E}$ des ensembles de combinaisons $[\mathcal{E}]_{min}$ et $[\mathcal{E}]_{max}$ tels que : $[\mathcal{E}]_{min}$ possède un représentant par connexion et permet de tracer un diagramme lisible,

$[\mathcal{E}]_{max}$ représente la multitude de combinaisons associée à chaque connexion de la structure.

Commençons par $[\mathcal{E}]_{min}$. Notre définition de la relation ≈ n'exclut pas des connexions telles que C = { {*ab,d*}, {*bc,d*}, {*ac,d*} }. Pour une telle connexion, il est impossible de définir un représentant minimal, car *d* n'est finalement connecté ni à *a*, ni à *b*, ni à *c*. Nous ajoutons donc une condition : un ensemble de combinaisons $\mathcal{E}$ est *bien fondé* si toutes ses connexions sont bien fondées ; une connexion C est *bien fondée* si C contient une *combinaison minimale*, c'est-à-dire s'il existe une combinaison {A,B} dans C telle que, pour toute combinaison {A',B'} dans C, on ait A ⊆ A' et B ⊆ B'.[8] Si $\mathcal{E}$ est bien fondé, on définit alors $[\mathcal{E}]_{min}$ comme l'ensemble des combinaisons minimales de $\mathcal{E}$. On a bien sur $[\mathcal{E}]_{min} \subseteq \mathcal{E}$.

Pour définir $[\mathcal{E}]_{max}$, nous devons étendre à toute structure la notion de catena introduite section 7 pour les arbres de dépendance. Pour commencer, appelons Unités($\mathcal{E}$) l'ensemble des unités qui apparaissent dans les combinaisons de $\mathcal{E}$. $\mathcal{E}$ définit une relation binaire sur Unités($\mathcal{E}$) que nous notons □ : A□B ssi {A,B} ∈ $\mathcal{E}$. Dans le cas où $\mathcal{E} = C_{min}$(D) avec D un arbre de dépendance, Unités($\mathcal{E}$) est un ensemble de mots et deux mots sont en relation par □ s'ils sont liés par une dépendance. On peut maintenant définir Catenae($\mathcal{E}$) comme l'ensemble des unités qui sont des catenae pour la relation □ sur Unités($\mathcal{E}$), c'est-à-dire des unités obtenues en réunissant des unités liées les unes aux autres par □. Dans le cas où $\mathcal{E} = C$(T) avec T un arbre de constituants binaire, Unités($\mathcal{E}$) est l'ensemble des constituants de T et Catenae($\mathcal{E}$) = Unités($\mathcal{E}$), car chaque constituant ne peut se combiner qu'avec un seul autre constituant (en raison de la binarité) et le résultat donne un constituant.

Nous considérons une deuxième propriété de bonne formation de nos ensembles de combinaisons, qui est une forme de connexité : nous dirons que $\mathcal{E}$ est *connexe* si toute unité A de $\mathcal{E}$ non élémentaire est un catena d'unités strictement incluses dans A. Cette condition assure que la structure soit suffisamment fine pour que toute unité soit décomposable jusqu'aux unités élémentaires. Nous imposons une troisième propriété de bonne formation qui est l'*acyclicité* de $\mathcal{E}$: la relation □ associée à $\mathcal{E}$ doit être acyclique, car sinon la relation ≈ ne serait pas une relation d'équivalence sur $[\mathcal{E}]_{max}$. En effet, supposons que $\mathcal{E}$ contienne {*a,b*}, {*b,c*}, {*a,c*}, alors *ab* est un catena, donc {*ab,c*} ∈ $[\mathcal{E}]_{max}$. Mais {*a,c*} ≈ {*ab,c*}, {*ab,c*} ≈ {*b,c*} et {*a,c*} ≉ {*b,c*}, ce qui montre la non-transitivité de ≈.

Comme dans le cas des arbres de dépendance, on associe à tout ensemble de combinaisons $\mathcal{E}$ connexe et acyclique l'ensemble de combinaisons $[\mathcal{E}]_{max}$ défini par :

$[\mathcal{E}]_{max}$ = Combi(Catenae($\mathcal{E}$)).

---

8 Il n'est pas nécessaire d'imposer que la combinaison minimale de C appartienne à C. On pourrait accepter des connexions telles que C = { {*ab,c*}, {*b,cd*} }, qui induit la combinaison minimale {*b,c*}.

La connexité et l'acyclicité assure que ≈ est bien une relation d'équivalence et que $\mathcal{E} \subseteq [\mathcal{E}]_{max}$.

On peut vérifier que $[C_{min}(D)]_{max} = C_{max}(D)$ et $[C_{max}(D)]_{min} = C_{min}(D)$ pour tout arbre de dépendance et que $[C(T)]_{min} = [C(T)]_{max} = C(T)$ pour tout arbre de constituants binaire. On peut généraliser ce type de propriété : pour tout ensemble de combinaisons bien fondé, connexe et acyclique, on a :

$[[\mathcal{E}]_{min}]_{max} = [\mathcal{E}]_{max}$

$[[\mathcal{E}]_{max}]_{min} = [\mathcal{E}]_{min}$.

Les deux derniers résultats découlent directement de nos propriétés de bonne formation : $[[\mathcal{E}]_{min}]_{max} = [\mathcal{E}]_{max}$, car, en raison de la connexité, toute unité de $\mathcal{E}$ peut être reconstruite à partir de sous-unités qui se combinent entre elles, et en particulier d'unités appartenant à des combinaisons minimales en raison de la bonne fondation des connexions. Quant à $[[\mathcal{E}]_{max}]_{min} = [\mathcal{E}]_{min}$, cela découle trivialement de la bonne fondation des connexions.

Nous allons donner un premier exemple d'utilisation des ensembles minimaux et maximaux. Considérons $\mathcal{E}3 = C_{max}(D1) \cap C_{max}(D2)$. Considérer l'ensemble $\mathcal{E}3$ revient à sous-spécifier l'analyse afin de ne pas prendre les décisions qui rendent $C_{max}(D1)$ et $C_{max}(D2)$ incompatibles. En l'occurrence, ces décisions sont de savoir qui du déterminant ou du nom est la tête du groupe *le chat* et qui de l'auxiliaire ou du verbe lexical gouverne le sujet. Si l'on garde pour chaque connexion de $\mathcal{E}3$ la plus petite combinaison, on obtient :

$$[\mathcal{E}3]_{min} = \{ \{le, chat\}, \{a, vu\}, \{le\ chat, a\ vu\}, \{vu, Zoé\} \} = C_{min}(S3)$$

où S3 est la structure à la Tesnière introduite dans la section 3 (figures 3 et 7). La structure S3 est une structure qui sous-spécifie la connexion subjectale en liant *le chat* à *a vu* et qui est ainsi la structure la plus fine qui soit compatible aussi bien avec D1 que D2. S3 est donc une sorte de supremum de D1 et D2. Nous allons formaliser cela.

8. Ordre et équivalence entre ensembles de combinaisons connexes

Nous avons vu que les $C_{max}(D)_{/\approx}$ et les $C_{min}(D)_{/\approx}$ ont le même nombre de connexions. D'une certaine façon, ils définissent la même structure, car leurs connexions sont deux à deux équivalentes. Ceci nous amène à étendre la relation ≈ sur les combinaisons aux connexions : les connexions C et C' sont *compatibles* par ≈ (ce que nous notons C ≈ C') ssi les combinaisons appartenant à C et C' sont compatibles par ≈. Ceci revient à dire que C ∪ C' est un ensemble de combinaisons équivalentes et définit également une connexion. On est tenté de dire que C, C' et C ∪ C' définissent plus ou moins la même connexion, mais il ne faut pas oublier que ces connexions restent quand même différentes car C et C' peuvent être de granularité différente (par ex., C = { {*chat, vu*} } et C' = { {*le chat, a vu Zoé*} }).

On peut maintenant définir une relation de compatibilité sur les ensembles de combinaisons que nous notons ≋. Deux ensembles de combinaisons $\mathcal{E}$ et $\mathcal{F}$ sont dits *compatibles*, ce que nous notons $\mathcal{E} \approxeq \mathcal{F}$, si leurs connexions sont deux à deux compatibles par ≈. Cela revient à dire que $\mathcal{E}$ et $\mathcal{F}$ définissent plus ou moins la même structure de connexion. La relation ≋ a la propriété suivante :

$\mathcal{E} \approxeq \mathcal{F}$ ssi ≈ est une relation d'équivalence sur $\mathcal{E} \cup \mathcal{F}$
et $\mathcal{E}/\approx$, $\mathcal{F}/\approx$ et $(\mathcal{E} \cup \mathcal{F})/\approx$ ont le même cardinal.

En effet, si une connexion C de $\mathcal{E}$ est compatible avec une connexion C' de $\mathcal{F}$, C ∪ C' forme une classe d'équivalence de ≈ sur E ∪ F et donc une connexion. On peut voir $\mathcal{E}$ et $\mathcal{F}$ comme des instanciations différentes des connexions définies par $(\mathcal{E} \cup \mathcal{F})/\approx$.

Comme nous l'avons montré dans la section 7, $C_{min}(D) \approxeq C_{max}(D)$, pour tout arbre de dépendance D. Par ailleurs, les ensembles de combinaisons définis par les arbres de dépendance sont compatibles avec de nombreux ensembles de combinaisons définis par des arbres de constituants. Par exemple, $C(T0) \approxeq C_{max}(D1)$, car $C(T0) \subseteq C_{max}(D1)$ et $C(T0)$ possède quatre connexions comme $C_{max}(D1)$. On a également $C(T0) \approxeq C_{min}(D1)$, comme on l'a vu à la section 6 en étudiant $\mathcal{E}4 = C_{min}(D1) \cup C(T0)$.

De même que ≈ n'est pas transitive sur les combinaisons, ≋ n'est pas transitive sur les ensembles de combinaisons. Par exemple, $C(T0)$ est compatible avec $C_{min}(D1)$ et $C_{min}(D2)$, alors que ces deux derniers ensembles ne sont pas compatibles entre eux, car $(C_{min}(D1) \cup C_{min}(D2))/\approx$ n'a pas le même cardinal que $C_{min}(D1)/\approx$ et $C_{min}(D2)/\approx$. En effet, ces deux derniers ensembles ont quatre éléments, tandis que $(C_{min}(D1) \cup C_{min}(D2))/\approx$ en a cinq, car les combinaisons c1 = {*chat, vu*} de $C_{min}(D1)$ et c2 = {*le, a*} de $C_{min}(D2)$ ne sont pas compatibles et appartiennent à deux connexions distinctes.

Si l'on considère à nouveau $\mathcal{E}3 = C_{max}(D1) \cap C_{max}(D2)$, on voit que l'on construit une structure compatible à la fois avec $C_{min}(D1)$ et $C_{min}(D2)$ et qui est plus « proche » de $C_{min}(D1)$ et $C_{min}(D2)$ que ne l'est $C(T0)$. $\mathcal{E}3$ contient une combinaison minimale {*le chat, a vu*} qui n'est pas élémentaire, et qu'on peut donc raffiner de différentes façons, ce qui nous donne, entre autres, les ensembles de combinaisons (incompatibles) $C_{min/max}(D1)$ et $C_{min/max}(D2)$. Dès qu'une combinaison est non élémentaire, on peut construire deux combinaisons plus fines incompatibles entre elles. Par exemple, pour {*ab,c*}, on a {*a,c*} et {*b,c*} qui sont plus fines et incompatibles entre elles. Si maintenant, nous prenons un ensemble de combinaisons $\mathcal{E}$ ayant {*ab,c*} comme combinaison minimale et que nous considérons $\mathcal{F} = \mathcal{E} \cup \{a,c\}$ et $\mathcal{F}' = \mathcal{E} \cup \{b,c\}$, nous aurons $\mathcal{E} \approxeq \mathcal{F}$ et $\mathcal{E} \approxeq \mathcal{F}'$, mais $\mathcal{F}$ et $\mathcal{F}'$ ne seront pas compatibles entre eux pour ≋. Ce que nous venons de montrer, c'est qu'en raffinant une des connexions de $\mathcal{E}$, on obtient un nouvel ensemble de combinaisons qui est compatible avec moins d'ensembles de combinaisons. En effet, $\mathcal{F}$ n'est pas compatible avec $\mathcal{F}'$, alors que $\mathcal{E}$ l'est. À l'inverse, on peut montrer

que $\mathcal{E}$ est compatible avec tous les ensembles compatibles avec $\mathcal{F}$, car $\mathcal{E} \subseteq \mathcal{F}$ et $\mathcal{E} \approxeq \mathcal{F}$.

Nous allons donner un sens précis à cette idée de finesse des connexions et de sur et sous-spécification. On peut, à partir de $\approxeq$, définir une relation d'ordre $\preceq$ telle que $\mathcal{E}$ soit *plus fin* que $\mathcal{F}$ ($\mathcal{E} \preceq \mathcal{F}$) si $\mathcal{E}$ est compatible avec moins d'ensembles que $\mathcal{F}$ et *équivalent* ($\mathcal{E} \equiv \mathcal{F}$) s'il est compatible avec les mêmes ensembles que $\mathcal{F}$. Nous introduisons la relation suivante sur les ensembles de combinaisons bien formés (c'est-à-dire bien fondés, connexes et acycliques) :

$\mathcal{E} \preceq \mathcal{F}$ ssi $\mathcal{E} \approxeq \mathcal{F}$ et $[\mathcal{F}]_{max} \subseteq [\mathcal{E}]_{max}$.

La relation $\preceq$ est un préordre (voir section 3) sur les ensembles de combinaisons connexes. Il est trivial de vérifier qu'elle est réflexive et transitive. Ce n'est pas un ordre, car la relation n'est pas antisymétrique. Nous définissons donc à partir de $\preceq$ la relation $\equiv$:

$\mathcal{E} \equiv \mathcal{F}$ ssi ($\mathcal{E} \preceq \mathcal{F}$ et $\mathcal{F} \preceq \mathcal{E}$) ssi $[\mathcal{E}]_{max} = [\mathcal{F}]_{max}$ ssi $[\mathcal{E}]_{min} = [\mathcal{F}]_{min}$.

La relation $\equiv$ est une relation d'équivalence. D'après ce nous avons vu à la section 7, $[\mathcal{E}]_{min} \equiv [\mathcal{E}]_{max}$ pour tout ensemble de combinaisons bien formé et $\mathcal{F} \equiv \mathcal{E}$ ssi $[\mathcal{E}]_{min} \subseteq \mathcal{F} \subseteq [\mathcal{E}]_{max}$. En particulier, $C_{min}(D) \equiv C_{max}(D)$ pour tout arbre de dépendance D.

Dans la figure 11, nous donnons un fragment de la relation $\preceq$. Il s'agit du treillis de relations pour toutes les structures plus fines qu'un arbre de constituants binaire T1 donné. (Par souci de lisibilité, nous n'avons pas mis les trois arbres de dépendance qui sont en dessous de la 4e structure sous T1 ; on peut facilement les déduire en regardant ceux qui sont liés à la 2e structure.) Chaque structure S' qui est sous une structure S précise une des connexions de S. Lorsqu'on part de l'arbre de constituants T1, on peut préciser soit la connexion mettant en jeu *a*, soit la connexion mettant en jeu *b*. La connexion mettant en jeu *a* peut être attribuée soit à *b* (1re structure), soit à *cd* (3e structure), tandis que la connexion mettant en jeu *b* peut être attribuée soit à *c* (2e structure), soit à *d* (4e structure). De ces quatre structures, celle qui donne le plus de possibilités est la 3e, où *cd* est connecté à *a* et *b*. On peut en effet connecter l'un des éléments, *a* ou *b*, à *c* ou *d*, tout en laissant l'autre connecté à *cd*. On obtient au final six arbres de dépendance plus fins que l'arbre de constituants T1. Remarquons encore que comme tout treillis d'ordre, le treillis de la figure 11 permet de visualiser facilement les infimums et supremums de deux structures. On peut, par exemple, pour chaque paire d'arbres de dépendance, voir la structure la plus fine les sous-spécifiant en considérant la structure à laquelle ils sont tous les deux reliés.

Comme le laisse supposer la figure 11, les éléments minimaux pour la relation $\preceq$ sont les $C_{min/max}(D)$ avec D arbre de dépendance, tandis que les éléments maximaux sont les $C(T)$ avec T arbre de constituants binaire.

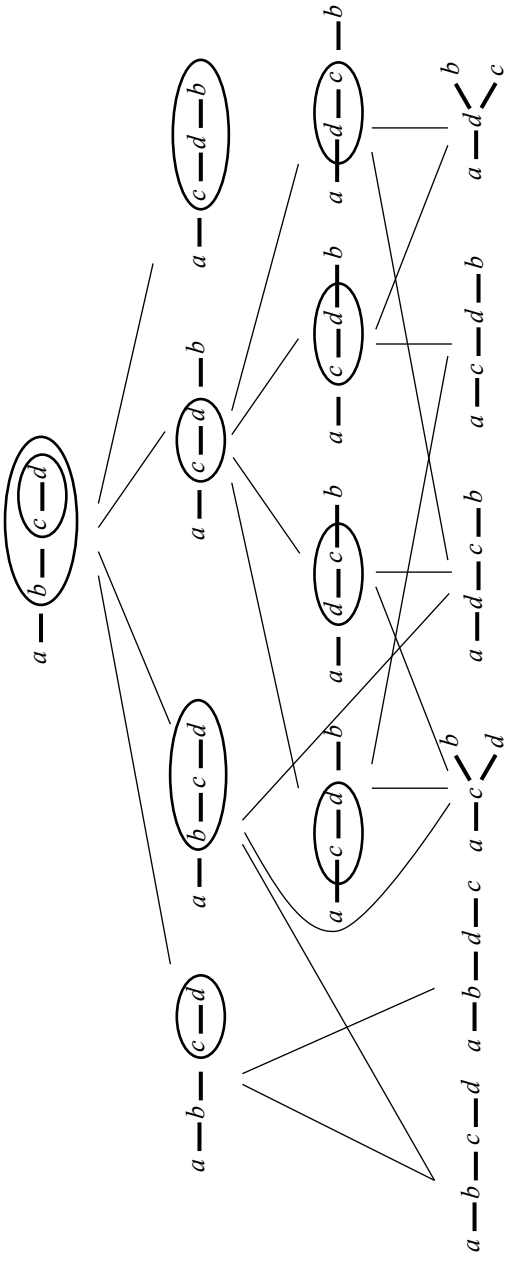

Figure 11. Treillis de la relation d'ordre $\preceq$ pour les structures plus fines qu'un arbre de constituants donné

Avant de montrer ces résultats, donnons-en une interprétation. Ces résultats signifient que les arbres de constituants binaires sont les structures qui spécifient le moins les connexions, alors que les arbres de dépendance sont celles qui les spécifient le plus. Entre les deux, il existe toute une gamme de structures comme le montre la figure 11. On peut aussi se reporter aux exemples présentés au début de cet article : la combinaison c0 = {*le chat, a vu Zoé*} de l'arbre de constituants T0 indique juste qu'il y a une combinaison entre deux grandes unités, alors que la stemma à la Tesnière S3 attribue cette connexion à une combinaison un peu plus fine {*le chat, a vu*} et qu'un arbre de dépendance comme D1 ou D2 attribue cette connexion à une combinaison encore plus fine (entre mots).

Les $C_{\min/\max}(D)$ sont bien les seuls éléments minimaux, puisque tout $\mathcal{E}$ minimal contient nécessairement des combinaisons minimales entre unités élémentaires, sinon les connexions correspondantes pourraient être raffinées. Vu les hypothèses de connexité et d'acyclicité, par ailleurs, les éléments minimaux sont bien des arbres de dépendance. Le fait que les $C(T)$ sont maximaux est une conséquence directe du fait que les $[C(T)]_{max} = C(T)$. Comme $C(T)$ possède une unique combinaison par connexion, il ne peut pas y avoir d'ensemble de combinaisons $\mathcal{E}$ bien formé avec $\mathcal{E} \subset C(T)$.

On peut montrer facilement que :

$$\mathcal{E} \vee \mathcal{F} \equiv [\mathcal{E}]_{max} \cap [\mathcal{F}]_{max}$$

$$\mathcal{E} \wedge \mathcal{F} \equiv [\mathcal{E}]_{max} \cup [\mathcal{F}]_{max} \equiv [\mathcal{E}]_{min} \cup [\mathcal{F}]_{min}$$

Il est possible que ces ensembles soient mal formés, ce qui signifie que $\mathcal{E}$ et $\mathcal{F}$ n'ont pas de supremum ou pas d'infimum. Par exemple, $C_{\min/\max}(D1)$ et $C_{\min/\max}(D2)$ sont des éléments minimaux non équivalents et n'ont donc pas d'infimum ($C_{\min}(D1) \cup C_{\min}(D2)$ contient un cycle). De même, deux $C(T)$ pour des arbres de constituants différents ne pourront pas avoir de supremum.

L'ordre $\preceq$ est déjà introduit dans Gerdes et Kahane (2013) sous une forme plus géométrique. L'objectif des auteurs est de montrer comment, à partir d'un ensemble d'unités syntaxiques assez riche, il est possible de construire une structure de connexion. Pour cela, les auteurs proposent à partir d'un ensemble d'unités U donné de construire n'importe quel arbre de constituants binaire dont les constituants appartiennent à U, puis de raffiner cet arbre jusqu'à obtenir une structure qui ne peut plus être raffinée. Cette structure peut en fait être construite directement : il s'agit de Combi(U). Ce que montre notre article, c'est que Combi(U) est l'infimum de tous les arbres de constituants sur U. Et si nous voulons faire un diagramme pour Combi(U), il suffit de prendre la combinaison minimale de chaque connexion, les connexions étant définies grâce à la relation d'équivalence $\approx$. Nous avons ainsi fourni un outil plus propre et direct pour définir une structure de connexion à partir d'un ensemble d'unités.

9. Granularité de la structure

Nous avons annoncé dans le titre de cet article la nécessité d'appréhender la structure en termes de connexions plutôt que d'unités. Il a malgré tout été beaucoup question d'unités, puisque les connexions se définissent en termes de combinaisons d'unités. Néanmoins, les unités n'ont pas plus de valeurs pour les connexions que les nombres entiers qui apparaissent dans la définition des nombres rationnels : de même que $1/2$ et $50/100$ représentent le même nombre, deux combinaisons équivalentes représentent la même connexion, indépendamment des unités qui les composent. Si l'on change la granularité de l'analyse, les unités élémentaires et les combinaisons minimales des connexions changent sans que cela ne change fondamentalement la structure (à l'exception des connexions qui sont plus fines que le grain et qui disparaîtront).

Nous allons formaliser cette question de granularité de la structure et de changement de grain. Le fait de voir la structure comme un ensemble de connexions permet de modifier facilement la granularité de la structure. Considérons un ensemble de combinaisons $\mathcal{E}$ défini sur un ensemble X d'unités élémentaires. On s'intéresse à un nouvel ensemble d'unités Y plus grossier que X, autrement dit un ensemble Y d'unités sur X. Nous voulons maintenant prendre Y comme ensemble d'unités élémentaires et regarder notre structure avec ce nouveau grain. Il suffit de considérer l'infimum de $\mathcal{E}$ et de l'ensemble Combi(Y) des combinaisons d'éléments de Y. Nous appelons cet infimum la *restriction* de $\mathcal{E}$ à Y. Par exemple, si Y est l'ensemble des chunks (qui sont des combinaisons particulières de mots, cf. Abney, 1991 ; Vergne, 2000), la restriction d'un arbre de dépendance à Y donnera un arbre dont les nœuds seront des chunks définis par l'arbre de dépendance original. Inversement, on peut considérer que la structure syntaxique se construit non pas sur les mots de la phrase, mais sur des unités plus fines, les lexèmes et les morphèmes flexionnels. Un arbre de dépendance traditionnel, qui est un arbre sur les mots, peut alors être vu comme la restriction aux mots d'une structure plus fine.

On voit à nouveau l'intérêt, lorsque l'on appréhende un arbre de dépendance, de voir cette structure syntaxique non pas comme une structure avec une combinaison par connexion ($C_{\min}(D)$), mais comme une structure avec de nombreuses instanciations possibles pour chaque connexion ($C_{\max}(D)$) et dont l'instanciation minimale dépend du grain considéré.

10. Conclusion

Nous allons essayer dans cette conclusion d'interpréter les résultats que nous avons présentés, au-delà de leur intérêt purement mathématique. Commençons par les résultats bruts de cet article. Nous avons introduit trois relations entre les structures de connexion : la relation de compatibilité $\approxeq$, la relation d'équivalence $\equiv$ et la relation d'ordre $\preceq$. Un arbre de dépendance D correspond à plusieurs structures de connexion équivalentes : si le diagramme lui-même représente $C_{\min}(D)$, c'est-à-

dire les combinaisons entre mots, c'est en tant que C_{max}(D) que la structure de dépendance doit être interprétée. On comprend alors que l'arbre de dépendance et les arbres de constituants sont différentes façons d'instancier les connexions de la structure de dépendance (voir figure 12).

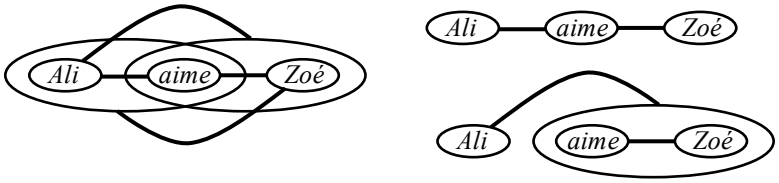

Figure 12. Une structure de dépendance C_{max}(D) et deux instanciations, par un arbre de dépendance et par un arbre de constituants

Arbres de dépendance et arbres de constituants sont des structures de connexion compatibles, mais pas équivalentes. L'arbre de dépendance instancie les connexions par des combinaisons minimales, des combinaisons entre mots, et l'arbre de constituants par des combinaisons maximales, des combinaisons entre constituants immédiats. On peut les voir comme deux modes de représentation de la structure syntaxique. Nous considérons qu'aucune des deux représentations ne correspond à la structure syntaxique, mais que ces représentations constituent deux vues différentes de la structure, avec des instanciations différentes des connexions. C'est ce qui nous permet de supposer que les connexions sont des classes d'équivalence de combinaisons, qui contiennent aussi bien les combinaisons maximales considérées par les arbres de constituants que les combinaisons minimales considérées par les arbres de dépendance.

Nous n'avons pas encore parlé du rôle joué par la notion de *tête* d'une unité (Bloomfield, 1933 ; Hudson 1984 ; Mel'čuk, 1988 ; Kahane 2001). C'est une notion complémentaire à la notion de connexion, qui permet de raffiner les combinaisons. Dès que l'on a une combinaison {*a,bc*} et que l'on sait que *a* est la tête de *abc* et *b* est la tête de *bc*, on peut postuler une combinaison {*a,b*} (Gerdes et Kahane 2013). C'est par cette méthode qu'à partir d'un arbre de constituants avec tête, on peut induire un arbre de dépendance (Lecerf, 1961). Cette méthode peut s'appliquer à n'importe quelle structure de connexion intermédiaire entre un arbre de constituants et un arbre de dépendance. Ceci a des conséquences immédiates sur la stratégie pour construire la structure syntaxique, aussi bien du point de vue du TAL que du point de vue de l'enseignement de la syntaxe. Cela signifie que, pour construire la structure, il suffit de construire n'importe quelle structure qui contienne au moins une instanciation de chaque connexion, puis de raffiner, si cela est possible, les différentes connexions en utilisant la notion de tête, afin d'obtenir la structure la plus fine possible.

Du point de vue cognitif, nous postulons que lors de l'analyse d'une phrase, les connexions sont instanciées par des combinaisons particulières portant sur des unités particulières et que ces instanciations peuvent être différentes d'une situation à

l'autre et ne correspondent *a priori* ni à un arbre de constituants, ni à un arbre de dépendance. Nous pensons que la prosodie (y compris la prosodie silencieuse du lecteur) joue un rôle important et que les unités prosodiques sont des candidats incontournables de cette instanciation (voir Steedman (2014) pour une approche similaire dans le cadre des grammaires catégorielles). À notre avis, du point de vue du TAL, cela a des conséquences importantes et signifie que les algorithmes de parsing pourraient prendre en compte le fait que l'on ne cherche pas à construire une instanciation particulière de la structure syntaxique (un arbre de constituants ou un arbre de dépendance en l'état actuel des systèmes), mais à construire n'importe quelle instanciation de la structure de connexion, quitte à la raffiner ensuite.

Si l'on considère maintenant la génération de texte, ce que nous combinons, ce sont des constructions (locutions, constructions de phrase, etc.) correspondant aux différents éléments de sens que l'on veut exprimer. L'ensemble de ces combinaisons constitue ce que Mel'čuk (1988) nomme la *structure syntaxique profonde*. Les connexions entre ces unités lexicales et grammaticales porteuses de sens vont ensuite être raffinées, quand ces mêmes unités seront réalisées par des combinaisons de lexèmes et de morphèmes flexionnels. On peut donc voir la structure syntaxique profonde comme une instanciation particulière de la structure syntaxique de surface, instanciation où sont regroupées ensemble les éléments formant ensemble une même unité sémantique élémentaire. Il existe encore un domaine du TAL qui considère des instanciations particulières de la structure syntaxique : la traduction automatique. En effet, les stratégies actuelles basées sur des mémoires de traduction consistent à rechercher les plus grandes sous-unités d'une phrase dont on a une traduction et à combiner leurs traductions. Ceci revient à instancier les connexions d'une manière particulière, les unités considérées n'étant généralement pas des constituants.

Revenons sur la nature même de la structure syntaxique. Il est possible que certaines connexions ne puissent pas être instanciées par une combinaison entre mots. Par exemple, considérer que, dans la phrase *le chien dort*, l'unité *le chien* possède une tête bien identifiée et que l'on puisse raffiner la combinaison {*le chien, dort*} de manière unique est probablement un biais de l'analyse par des arbres de dépendance, qui oblige à instancier chaque connexion par une combinaison entre mots ou, ce qui revient finalement au même, à choisir une tête pour chaque unité. En fait, par leur côté extrême, les deux modes de représentation, dépendance et constituance, présentent des biais. Le biais pour les arbres de constituants est la *stratification* : il faut à chaque étape de l'analyse en constituants immédiats décider quelles sont les deux unités qui se connectent, même lorsqu'il y a plusieurs connexions disponibles. Il n'est pas possible avec un arbre de constituants de dire simplement que, dans *Ali aime Zoé*, *aime* se combine avec *Ali* et *Zoé*. On doit stratifier, c'est-à-dire traiter les connexions dans un certain ordre (en considérant par exemple que *Ali* se combine avec *aime Zoé* qui est lui-même la combinaison de *aime* et *Zoé*). Il en découle qu'il y a autant d'arbres de constituants associés à une structure de connexion donnée que de façons de stratifier l'ensemble des connexions.

Par ailleurs, les analyses se placent généralement à un niveau de granularité particulier, celui des mots ou des unités morphosyntaxiques (lexèmes et morphèmes flexionnels). Là encore il s'agit d'instanciations particulières des connexions. Il est important de concevoir que la même structure syntaxique peut être envisagée à différents niveaux de granularité. Nous pensons de surcroît que les locuteurs manipulent des unités de différents niveaux aussi bien lorsqu'ils produisent que lorsqu'ils analysent des énoncés. Nous avons montré que notre formalisation des connexions ne fait aucune hypothèse sur la nature des unités mises en jeu lors de leur réalisation. Nous défendons l'idée que l'on peut faire de la syntaxe sans poser *a priori* la question des unités, laquelle est très délicate, tant il est difficile de définir des notions comme celles de mot ou de phrase. De ce point de vue, une définition qui dirait que la syntaxe est l'étude de l'organisation des mots au sein de la phrase nous semble à rejeter totalement. Pour nous, la syntaxe est avant tout l'étude des combinaisons (libres, régulières) entre signes linguistiques, sans préjuger du niveau de granularité de ces signes.

Remerciements. Je remercie François Lareau, Nicolas Mazziotta et les 3 relecteurs pour leurs nombreuses remarques qui m'ont permis, je l'espère, de rendre le texte plus facile à lire.

Bibliographie

Abney S. P., *The English noun phrase in its sentential aspect*. Thèse de doctorat. Massachusetts Institute of Technology, 1987.

Abney S. P., « Parsing by chunks », in Berwick R. C., Abney S. P., Tenny, C. (éds), *Principle-based parsing* [*Computation and psycholinguistics*, 44], Springer, Dordrecht, 1991, 257-278.

Beauzée N., « Régime », in Denis Diderot & Jean Le Rond D'Alembert J. (eds.), *Encyclopédie ou Dictionnaire raisonné des sciences, des arts et des métiers*, vol. 14, 1765, 5-11.

Bourbaki N., *Théorie des ensembles*, Hermann, 1939.

Bloomfield L., *Language*, The University of Chicago Press, 1933.

Chomsky N., « Three models for the description of language », *IRE Transactions on information theory*, *2*(3), 1956, 113-124.

Chomsky N., *Syntactic Structures*, MIT Press, Cambridge, 1957.

Debili F., *Analyse syntaxico-sémantique fondée sur une acquisition automatique de relations lexicales-sémantiques*, Thèse de doctorat d'état, Université Paris Sud, Orsay, 1982.

de Marneffe M.-C., Manning C. D., « The Stanford typed dependencies representation », *Proceedings of the Workshop on Cross-framework and Cross-domain Parser Evaluation*, COLING, 2008.

Gerdes K., Kahane S., « Defining dependency (and constituency) », in K. Gerdes, E. Hajičová, L. Wanner (éds.), *Computational Dependency Linguistics*, IOS Press , 2013.

Gladkij A. V., « On describing the syntactic structure of a sentence » (en russe avec résumé en anglais), *Computational Linguistics, 7*, Budapest, 1968, 21-44.

Hudson R. A., *Word grammar*, Oxford: Blackwell, 1984.

Hudson R. A., *Language networks: The new word grammar*, Oxford University Press, 2007.

Kahane S., « Bubble trees and syntactic representations », in *Proceedings of the 5th conference on Mathematics of Language (MoL)*, 1997, 70-76.

Kahane S., « Grammaires de dépendance formelles et théorie Sens-Texte », tutoriel, *Actes TALN 2001*, *vol. 2*, 2001, 17-76.

Kahane S., Mazziotta N., « Syntactic Polygraphs. A Formalism Extending Both Constituency and Dependency », *Proceedings of the 14th Meeting on the Mathematics of Language (MoL)*, Association for Computational Linguistics, 2015, 152-164.

Kahane S., Osborne T., « Translators' introduction », in L. Tesnière, *Elements of structural syntax*, John Benjamins, 2015, ixxx-lxiii (49 p.).

Lecerf Y., « Une représentation algébrique de la structure des phrases dans diverses langues naturelles », *Comptes Rendus de l'Académie des Sciences*, *252*(2), 1961, 232-235.

Mel'čuk I., *Dependency syntax : Theory and practice*, SUNY, New York, 1988.

Mazziotta N., Kahane S., « To what extent is Immediate Constituency Analysis dependency-based? A survey of foundational texts », *Proceedings of the Fourth International Conference on Dependency Linguistics (Depling)*, 2017, 116-126.

Nida E., *A synopsys of English Syntax*. Mouton and Co., London, The Hague, Paris, 2e édition, 1966.

Nivre J., De Marneffe M.-C., Ginter F., Goldberg Y., Hajic J., Manning C. D., ..., Tsarfaty R., « Universal Dependencies v1: A Multilingual Treebank Collection », *Proceedings of LREC*, 2016.

Osborne T., Putnam M., Gross T., « Catenae: Introducing a novel unit of syntactic analysis », *Syntax*, 15(4), 2012, 354-396.

Steedman M., « The Surface Compositional Semantics of English Intonation », *Language*, 90, 2014, 2-57.

Tesnière L., *Eléments de syntaxe structurale*, Klincksieck, Paris, 1959.

Vergne J., *Étude et modélisation de la syntaxe des langues à l'aide de l'ordinateur - Analyse syntaxique automatique non combinatoire*, Thèse d'HDR, Université de Caen, 2000.

Les méthodes « apprendre à chercher » en traitement automatique des langues : un état de l'art

Elena Knyazeva — Guillaume Wisniewski — François Yvon

LIMSI, CNRS, Univ. Paris-Sud, Université Paris-Saclay,
Campus universitaire bât 508, Rue John von Neumann
F - 91405 Orsay cedex
{knyazeva,wisniews,yvon}@limsi.fr

RÉSUMÉ. *L'apprentissage structuré est au fondement des méthodes modernes d'apprentissage automatique pour le traitement automatique des langues (TAL). Dans cet article, nous étudions une famille d'algorithmes d'apprentissage structuré, les algorithmes de la famille « apprendre à chercher », qui diffèrent fondamentalement des méthodes classiques telles que les champs markoviens aléatoires, et permettent donc de mettre en évidence certains des compromis de l'apprentissage structuré en TAL. Nous présentons également un panorama des applications de ces techniques en TAL, en discutant les bénéfices découlant de leur utilisation.*

ABSTRACT. *Structured prediction lies at the heart of modern Natural language Processing (NLP). In this paper, we study a specific family of structured learning algorithms, loosely refered to as "Learning-to-search" algorithms. They differ in several important ways from more studied methods such as Conditional Random Fields, and their study highlights several important trade-offs of structured learning for NLP. We also present an overview of existing applications of these techniques to NLP problems and discuss their potential benefits.*

MOTS-CLÉS : *traitement automatique des langues, apprentissage structuré, apprendre à chercher.*

KEYWORDS: *Natural Language Processing, Structured learning, Learning to search.*

1. Introduction

L'utilisation de méthodes d'apprentissage supervisé est l'approche de référence en traitement automatique des langues (TAL), et permet d'obtenir les meilleures performances aussi bien sur des tâches *intermédiaires* d'analyse linguistique (étiquetage en parties du discours, analyse syntaxique, identification de relations de coréférence, détection d'entités nommées, etc.), que pour des tâches *finalisées* (traduction automatique, dialogue, extraction d'information, etc.).

La mise en œuvre de ces méthodes repose sur la modélisation, sur un corpus d'apprentissage, de corrélations statistiques entre des descripteurs et une ou plusieurs variables d'intérêt, permettant ensuite le calcul (l'inférence) des valeurs de ces variables lorsqu'elles ne sont pas observées. Cette stratégie est rendue particulièrement difficile en TAL par (a) le besoin de modéliser des phénomènes à partir d'observations parfois très éparses ; (b) le caractère *structuré* des entités linguistiques (mots, phrases, documents), qui implique des dépendances souvent complexes, et potentiellement à longue distance, entre les variables que l'on doit prédire. Un exemple typique de (b) est l'accord grammatical entre un sujet et le verbe principal, qui contraint les deux termes à partager des traits communs (personne et nombre en français) indépendamment de leurs positions respectives dans une phrase, qui peuvent être arbitrairement éloignées. Prendre en compte ces dépendances demande d'employer des techniques *d'apprentissage structuré* (Bakir *et al.*, 2007 ; Smith, 2011), qui sont donc les techniques de référence pour de nombreuses tâches : *parsing* stochastique, alignement de mots, détection de coréférence, extraction d'information, traduction automatique, etc.

Toute méthode d'apprentissage structuré doit faire face aux questions (a) et (b). Une réponse pragmatique consiste à ne prendre en compte que des dépendances « proches »[1], par exemple en bornant *a priori* les dépendances au voisinage immédiat selon une hypothèse markovienne à l'ordre 1 ou 2. Utiliser des hypothèses simplificatrices rend l'estimation moins sensible au caractère épars des données. Cela permet également l'utilisation de méthodes d'estimation et d'inférence reposant sur *des algorithmes exacts* de complexité raisonnable tels que l'algorithme de Viterbi (1967) pour l'inférence dans les modèles de Markov cachés.

Une alternative consiste à utiliser des contextes plus larges, en renonçant à l'exactitude de l'inférence, remplacée pour la circonstance par des techniques de *recherche gloutonne*. C'est le cas des méthodes « apprendre à chercher »[2] (L2S[3]) (Daumé *et al.*, 2009 ; Ross *et al.*, 2011) que nous étudions ici, avec pour ambition d'en donner une présentation compacte et précise. Ces méthodes sont intéressantes car, outre leurs performances souvent équivalentes à celles des méthodes standard comme les champs markoviens conditionnels (Tellier et Tommasi, 2011), elles jettent un nouvel éclai-

1. « Proche » n'est pas nécessairement entendu en relation avec l'ordre linéaire des mots et peut aussi s'interpréter au travers les relations de domination dans un arbre.
2. La terminologie est un peu fluctuante et l'on parle également d'apprentissage par imitation. Nous nous en tenons à « apprendre à chercher », qui nous semble moins ambigu.
3. Pour *Learning to Search*.

rage sur les compromis de l'apprentissage structuré en TAL. L'idée principale de ces techniques consiste à (a) reformuler l'inférence d'une structure linguistique sous la forme d'une séquence de décisions locales, chaque séquence de décisions définissant une trajectoire dans l'espace (combinatoirement grand) des structures possibles ; (b) apprendre à réaliser ces décisions locales en utilisant *un classifieur sensible aux coûts* (c'est-à-dire un classifieur qui prend en compte le fait que les différentes erreurs de classification peuvent impliquer différents coûts (Elkan, 2001)) qui intègre toutes les décisions prises dans le passé. L'enjeu est alors de minimiser les erreurs du classifieur local, afin de maximiser les chances de construire une structure entièrement correcte. Il importe donc que les exemples d'apprentissage du classifieur soient le plus proches possibles des contextes locaux qui seront effectivement visités pendant l'inférence ; le choix optimal ne consistant pas nécessairement à utiliser ceux qui découlent des annotations de référence.

Cet article est organisé comme suit : nous posons, à la section 2 un certain nombre de concepts nécessaires à la bonne exposition des principales méthodes d'apprendre à chercher de la littérature, en particulier les notions de *politique expert et tuteur* ; nous détaillons ensuite, à la section 3, les deux principales familles d'algorithmes (AggreVaTe et SEARN) de la littérature, qui reposent toutes les deux sur la notion *d'itérations de politique*. Au terme de ces deux sections théoriques, la section 4 présente un certain nombre d'applications au TAL de ces techniques, en essayant de mettre en évidence leur intérêt par rapport à l'état de l'art. Nous concluons à la section 5 en présentant quelques problèmes ouverts et pistes de développement pour ces techniques. Une annexe, disponible en ligne, détaille les preuves esquissées dans le corps du texte[4].

2. Apprendre à chercher : un autre point de vue sur l'apprentissage structuré

2.1. *Apprentissage structuré : quelques définitions*

2.1.1. *Apprentissage structuré*

Dans cette section, nous posons les principales notions nécessaires à la compréhension des méthodes détaillées dans la section 3. Comme pour tout problème d'apprentissage supervisé, nous observons un ensemble de couples associant une entrée $x \in X$ et une sortie de référence $y \in Y$, supposés échantillonnés sous une distribution $\widetilde{D}$ inconnue. Pour fixer les idées, nous utiliserons en guise d'illustration la tâche de traduction. Pour cette tâche, x et y sont respectivement la phrase à traduire et sa traduction de référence. Nous supposons également donnée une fonction de perte $L(y, \hat{y})$, qui évalue l'erreur commise lorsque l'on prédit $\hat{y}$ au lieu de la référence y. La métrique standard pour mesurer la qualité d'une traduction est le score BLEU (Papineni *et al.*, 2002)[5] ; la fonction de perte peut être, par exemple, l'opposé de ce score.

4. `perso.limsi.fr/yvon/publications/sources/Knyazeva19L2S-annexe.pdf`

5. BLEU n'est pas une métrique bien adaptée à l'évaluation de phrases isolées, et il faut plutôt utiliser des variantes de ce score.

Le point de vue standard en apprentissage structuré (Smith, 2011) considère un ensemble paramétrique de fonctions f_θ permettant d'associer x à $y = f_\theta(x)$ et d'optimiser les paramètres θ en minimisant la perte moyenne sur un ensemble d'apprentissage. Le point de vue des méthodes « apprendre à chercher » est différent : le focus est mis sur le processus de construction de la sortie y, envisagé à travers l'exploration de l'espace de toutes les sorties possibles. Pour apprendre à bien prédire y, il importe de savoir explorer l'espace de recherche et d'y prendre les bonnes décisions. Cet objectif peut sembler moins directement lié à la résolution du problème initial, mais il a l'avantage de se formaliser comme une suite de décisions simples, qu'il est possible d'apprendre à réaliser avec des outils génériques et peu coûteux de classification. Nous introduisons quelques notions de base, avant de formaliser la relation entre le problème structuré et la séquence de problèmes de classification associée.

2.1.2. *Espace de recherche*

Nous considérons des problèmes d'apprentissage *structuré*, pour lesquels l'ensemble des sorties y possibles pour un x donné est un ensemble combinatoire, représenté par un graphe. Formellement, pour une entrée x et sa sortie de référence $y \in Y$, l'espace de recherche $G(x, y) = \langle S,\, A,\, S_f,\, s_i,\, r\rangle$ est composé de[6] :

– S est l'ensemble (fini) des états possibles ;

– A est l'ensemble (fini) des actions possibles ;

– $S_f \subset S$ est l'ensemble des états finaux ;

– $s_i \in S$ est un état initial (unique pour chaque espace de recherche) ;

– $r(s, a)$ est la *récompense* associée à l'action a dans l'état s.

La figure 1 représente un espace de recherche pour la tâche de traduction. Chaque état correspond à une traduction partielle produite ; une action consiste à choisir un ou plusieurs mots pour continuer la traduction partielle. L'état initial correspond à la traduction vide, et toute traduction complète est un état final. La définition des récompenses sera explicitée ci-dessous.

On notera A_s l'ensemble des actions réalisables dans l'état s et $s' = s \circ a$ l'opération d'extension qui sélectionne l'action $a \in A_s$ dans l'état s afin d'atteindre l'état s'. Dans le cadre de ce travail, cette fonction est supposée déterministe, ce qui signifie que la connaissance de l'état et de l'action est suffisante pour déterminer l'état d'arrivée. L'espace de recherche forme un graphe sans cycle à l'exception d'une boucle sur chaque état final, dont la fonction sera clarifiée dans la section 2.2.1[7].

On appelle alors *chemin de longueur* t une suite $\alpha = (s_1, a_1) \dots (s_t, a_t)$ de couples dans $S \times A$, telle que $s_{t'+1} = s_{t'} \circ a_{t'}$ pour $1 \leq t' \leq t - 1$. Un chemin

6. Ceci est une simplification du concept de processus de décision markovien (Puterman, 1994), dans laquelle l'état d'arrivée est une fonction déterministe de l'état de départ et de l'action effectuée.

7. Un tel graphe est généralement appelé treillis, bien que, contrairement à la définition standard, il contienne plusieurs états finaux.

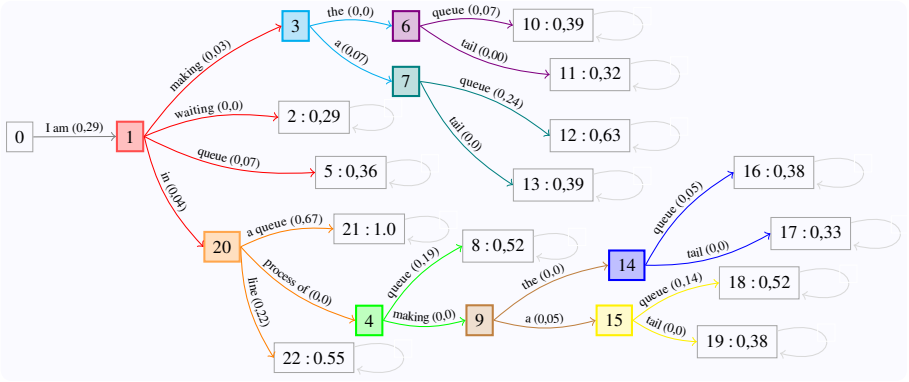

Figure 1. *Exemple d'un espace de recherche de traductions en anglais de la phrase « Je suis en train de faire la queue ». Les états pour lesquels plusieurs actions sont possibles sont colorés.*

est *complet* si $s_1 = s_i$ et $s_t \circ a_t \in S_f$; dans le cas contraire, on parlera d'un *chemin partiel*. La séquence d'actions $a_1 \dots a_t$ composant un chemin complet α peut être vue comme une suite d'instructions permettant de construire la sortie $\hat{y}_\alpha$ [8].

La *récompense cumulée* d'un chemin complet est la somme des récompenses associées aux actions le long de ce chemin :

$$R(\alpha) = \sum_{(s,a)\in\alpha} r(s,a),$$

et correspond à la qualité de la solution $\hat{y}_\alpha$. La récompense est définie en relation avec la fonction de perte de la manière suivante : la différence entre les récompenses cumulées de deux chemins est égale à l'opposé de la différence de pertes associées aux solutions correspondantes. Pour deux chemins α_1 et α_2, on a donc :

$$L(y, \hat{y}_{\alpha_1}) - L(y, \hat{y}_{\alpha_2}) = -(R(\alpha_1) - R(\alpha_2)).$$

Dans notre exemple, les états finaux sont équipés des scores 2-BLEU[9] des traductions produites ; ces scores correspondent aux récompenses cumulées le long du chemin menant dans les états finaux. Les récompenses locales sont définies de manière que la somme des récompenses sur chaque chemin complet soit égale au score de la traduction produite. Leurs valeurs sont données pour chaque action entre parenthèses.

8. Pour simplifier les démonstrations, on notera T la longueur minimale telle que pour chaque espace de recherche simulé sous $\widetilde{D}$, tous les chemins de longueur T ou plus soient complets.
9. On utilise généralement le score 4-BLEU, qui combine les précisions $1, 2, 3$, et 4-grammes (Papineni *et al.*, 2002). Nous utilisons ici 2-BLEU, qui est mieux adapté à l'évaluation de phrases isolées, pour lesquelles les précisions d'ordre supérieur sont souvent nulles.

2.1.3. *Politique et notions associées*

Une *politique* est une fonction déterminant l'action à choisir dans un état donné. Cette fonction peut être déterministe ou stochastique. Dans le premier cas, on parle de *politique déterministe* $\tau : S \mapsto A$, associant à chaque état $s \in S$ une unique action $a \in A_s$. L'application de la politique τ depuis l'état initial jusqu'à un état final produit un chemin α_τ, qui correspond à la sortie $\hat{y}_{\alpha_\tau}$. Un exemple de politique déterministe pour l'espace de recherche de la figure 1 est :

$$\check{\tau} = \{1 \to 5, 3 \to 7, 4 \to 8, 6 \to 10, 7 \to 12, 9 \to 15, 14 \to 16, 15 \to 18, 20 \to 4\}$$

Cet ensemble correspond aux choix des actions pour chaque état où plusieurs actions sont possibles. L'application de $\check{\tau}$ depuis l'état initial produit la traduction « I am queue ». $\check{\tau}$ peut aussi être appliquée depuis un état interne : son application depuis l'état 9 produit la traduction « I am in process of making a queue ».

Une politique stochastique est une fonction stochastique $\pi : S \mapsto A$ qui, à chaque état $s \in S$, associe une action selon une certaine distribution de probabilité sur A_s. Dans ce cas, le chemin produit, ainsi que la sortie structurée, sont également stochastiques. Nous noterons la distribution correspondante sur les chemins par $\widetilde{\alpha}_\pi$. Les politiques déterministes étant des cas particuliers de politiques stochastiques, on supposera dans la suite que, sauf mention explicite, toute politique π est stochastique.

2.1.3.1. Qualité de la politique

La qualité de la politique π se mesure à l'aune de la qualité de la sortie structurée qu'elle produit. Afin de mettre en évidence cette relation, nous utiliserons la même notation pour exprimer ces deux notions. Ainsi, la qualité d'une politique déterministe est simplement la perte associée à la sortie structurée produite par cette politique :

$$L(\tau) = L(y, \hat{y}_{\alpha_\tau}).$$

La qualité d'une politique stochastique est la perte moyenne sur les sorties produites :

$$L(\pi) = \mathbb{E}_{\alpha \sim \widetilde{\alpha}_\pi}[L(y, \hat{y}_\alpha)].$$

L'objectif de l'apprentissage est de trouver, à partir des données étiquetées, une politique qui minimise cette perte.

2.1.3.2. Utilisation de politique

Quelques notions supplémentaires pour manipuler les politiques sont nécessaires :

– $s \circ \pi^t$ est le résultat de l'application d'une suite de t actions choisies par la politique π depuis l'état s ; en particulier, $s \circ \pi = s \circ \pi^1$ pour une seule application de la politique. La politique π étant supposée stochastique, ce résultat l'est aussi (c'est une distribution sur les états pouvant être obtenus). *C'est le cas de tous les objets définis dans cette section lorsqu'une politique stochastique est utilisée ;*

– $s \circ \pi^*$ note le résultat de l'application d'un nombre arbitraire (entre 0 et T) d'actions choisies par la politique π dans l'état s.

Ces notions peuvent être généralisées au cas d'une distribution sur les états de départ $\widetilde{s}$. Par exemple, $\widetilde{s} \circ \pi$ est la distribution sur les états obtenue en appliquant l'action choisie par π sur la distribution $\widetilde{s}$ d'états de départ. On introduit maintenant des notions concernant les chemins produits à l'aide d'une politique π :

– $s \xrightarrow{\pi, t} \cdot$ est le chemin de t actions construit à l'aide de π à partir de l'état s ;

– $s \xrightarrow{\pi, \cdot} S_f$ est le chemin construit à l'aide de la politique π à partir de l'état s jusqu'à un état final (la longueur du chemin n'est pas fixée).

On peut également combiner plusieurs politiques pour construire un chemin complexe. $s \xrightarrow{\pi_1, t_1} \cdot \xrightarrow{\pi_2, t_2} \cdot \xrightarrow{\pi_3, \cdot} S_f$, est ainsi un chemin où les t_1 premières actions sont choisies par la politique π_1, les t_2 actions suivantes sont choisies par la politique π_2 et la suite du chemin jusqu'à l'état final est construite par la politique π_3.

Ces notions se généralisent aussi au cas d'une distribution d'états de départ.

2.1.3.3. Récompenses cumulées et fonctions de valeur

Nous étendons maintenant les notions de récompenses aux chemins. La *récompense cumulée* $R(\widetilde{\alpha})$ d'un chemin $\widetilde{\alpha}$ stochastique se définit par :

$$R(\widetilde{\alpha}) = \mathbb{E}_{\alpha \sim \widetilde{\alpha}}[R(\alpha)].$$

La *fonction de valeur* V *d'un état* s vis-à-vis de la politique π est définie par :

$$V(s; \pi) = R(s \xrightarrow{\pi, \cdot} S_f).$$

Elle représente la récompense cumulée depuis l'état s jusqu'à l'état final, lorsque toutes les actions sont choisies selon la politique π. Pour notre exemple, $V(0; \check{\tau}) = 0,36$ (ce qui correspond au score de la traduction « I am queue ») et $V(1; \check{\tau}) = 0,07$ (ce qui correspond au score de la même traduction moins la récompense associée à l'action effectuée pour arriver dans l'état 1).

La *fonction de valeur* Q *d'un couple* (*état* s, *action* $a \in A_s$) vis-à-vis de la politique π représente la récompense cumulée depuis l'état s jusqu'à l'état final sachant que la première action est a et que les actions suivantes sont choisies selon π :

$$Q(s, a; \pi) = R(s \xrightarrow{\cdot} s \circ a \xrightarrow{\pi, \cdot} S_f).$$

Dans notre exemple, $Q(1, 1 \rightarrow 20; \check{\tau}) = 0,23$ (le score de la traduction « I am in process of queue » moins la récompense de la première action $0 \rightarrow 1$).

Il est possible d'étendre ces fonctions au cas où l'état et l'action sont stochastiques, le cas important étant celui où l'action a est donnée par une politique stochastique. Soient π_1 et π_2 deux politiques quelconques, s distribué selon $\tilde{s}$. On a alors :

$$V(\tilde{s}; \pi_2) = \mathbb{E}_{s \sim \tilde{s}} \left[R(s \xrightarrow{\pi_2, \cdot} S_f) \right] \quad Q(\tilde{s}, \pi_1; \pi_2) = \mathbb{E}_{s \sim \tilde{s}} \left[R(s \xrightarrow{\pi_1, 1} \cdot \xrightarrow{\pi_2, \cdot} S_f) \right]$$

2.1.4. *L'expert et le tuteur*

L'*expert* $\pi_{\exp}$ est une politique qui permet, lorsqu'elle est exécutée de bout en bout, de construire la meilleure solution présente dans l'espace de recherche[10]. Dans notre cas, cette meilleure solution est la traduction de référence « I am in a queue ». Soit $\alpha^*(G)$ l'ensemble des chemins de l'espace de recherche G, alors pour la politique expert la propriété suivante est vérifiée :

$$\hat{y}_{\pi_{\exp}} = \underset{\alpha \in \alpha^*(G)}{\arg\min}\ L(y, \hat{y}_\alpha).$$

Dans ce travail, on suppose avoir accès à un expert plus puissant qui, *pour toute solution partielle*, permet de construire la meilleure solution complète possible[11]. C'est-à-dire que même si le début d'un chemin est construit à l'aide d'une autre politique, l'expert permet de trouver la meilleure manière de le poursuivre. Avec les notations introduites dans la section précédente, cet expert peut être exprimé par[12] :

$$\pi_{\exp}(s) = \underset{a \in A_s}{\arg\max}\, Q(s, a, \pi_{\exp}).$$

Ainsi, l'expert est la politique choisissant l'action qui conduira à la meilleure récompense cumulée, sachant que toutes les actions suivantes jusqu'à l'état final seront également choisies par la politique expert. Pour l'exemple de la figure 1 :

$$\check{\tau}_{\exp} = \{1 \to 20, 3 \to 7, 4 \to 8, 6 \to 10, 7 \to 12, 9 \to 15, 14 \to 16, 15 \to 18, 20 \to 21\}.$$

Cette politique suivie du début à la fin construit la traduction de référence « I am in a queue » ; appliquée depuis l'état 4, elle construit la meilleure traduction atteignable « I am in process of making a queue ». Certains algorithmes présentés dans ce travail nécessitent un expert capable d'agir de la meilleure manière possible étant donné non seulement le début du chemin, mais en sachant aussi que la suite du chemin sera construite par une autre politique. Un tel expert est appelé un *tuteur*. La politique tuteur peut être formalisée par :

$$\pi_{\mathrm{tut}}(s, \pi') = \underset{a \in A_s}{\arg\max}\, Q(s, a; \pi').$$

Ainsi, le tuteur choisit l'action impliquant la meilleure récompense cumulée sachant que toutes les actions suivantes jusqu'à l'état final seront choisies par la politique π'. La politique tuteur pour l'exemple figure 1 vis-à-vis de la politique $\check{\tau}$ est :

$$\check{\tau}_{\mathrm{tut}}(\cdot, \check{\tau}) = \{1 \to 3, 3 \to 7, 4 \to 8, 6 \to 10, 7 \to 12, 9 \to 15, 14 \to 16, 15 \to 18, 20 \to 21\}$$

10. Cette meilleure solution n'est pas nécessairement identique à la référence, puisqu'il est possible que celle-ci ne soit pas présente dans l'espace de recherche. Ce problème est documenté (par exemple) dans (Liang *et al.*, 2006 ; Sokolov *et al.*, 2013).

11. Cette idée rejoint la notion d'« oracle complet » présentée dans (Goldberg et Nivre, 2012).

12. Si plusieurs actions permettent d'obtenir la valeur Q maximale, alors la politique expert est non déterministe. Dans ce travail nous ne considérons par spécialement ce cas ; plus de détails peuvent être trouvés par exemple dans (Knyazeva *et al.*, 2015).

Cette politique est différente de la politique expert $\check{\tau}_{\text{exp}}$ seulement pour l'état 1 : en proposant l'action $1 \rightarrow 3$ plutôt que l'action $1 \rightarrow 20$ elle tient compte du fait que les actions suivantes seront prises avec la politique $\check{\tau}$ et que forcer l'action $1 \rightarrow 3$ dans ce cas permettra d'obtenir le score final $0,63$ au lieu de $0,52$.

2.1.5. *La perte de potentiel*

Dans ce travail on suppose qu'il est possible de calculer la perte immédiate apportée par une action a (toujours vis-à-vis d'une politique). On appelle cette perte immédiate la *perte de potentiel* :

$$l(s, a, \pi) = \max_{a' \in A_s} Q(s, a'; \pi) - Q(s, a; \pi)$$

pour tout état $s \in S$ et pour toute action $a \in A_s$ vis-à-vis de la politique π. Pour notre exemple, $l(1, 1 \rightarrow 3, \check{\tau}_{\text{exp}}) = 0,37$, ce qui correspond à la différence des scores de « I am in a queue » et « I am making a queue ») et $l(1, 1 \rightarrow 3, \check{\tau}) = 0$ ($1 \rightarrow 3$ est la meilleure action dans l'état 1 quand le reste des actions est effectué selon $\check{\tau}$).

Comme pour les fonctions V et Q, nous généralisons la perte de potentiel aux cas où états et actions sont stochastiques. Quand l'action stochastique est donnée par une politique stochastique π_1 et l'état s est représenté par une distribution $\widetilde{s}$, on obtient :

$$l(\widetilde{s}, \pi_1, \pi_2) = \mathbb{E}_{s \sim \widetilde{s}} \left[\max_{a' \in A_s} Q(s, a'; \pi_2) - Q(s, \pi_1; \pi_2) \right].$$

2.2. *Réduction de l'apprentissage structuré à un problème de classification sensible aux coûts*

Intuitivement, apprendre à chercher consiste à apprendre, pour chaque état de l'espace de recherche, à choisir localement l'action qui conduira à la meilleure sortie possible. Nous formalisons ici ce lien entre classification locale et apprentissage structuré.

2.2.1. *Fonction objectif*

Comme expliqué ci-dessus, suivre la politique expert sur la trajectoire complète permet d'obtenir la meilleure solution présente dans l'espace de recherche. Il est néanmoins rarement possible d'imiter parfaitement la politique expert. À défaut, l'objectif de l'apprentissage est alors de trouver la politique π permettant de construire des sorties conduisant à la plus petite perte $L(\pi)$ possible.

Lemme 2.1 *La perte d'une politique π se décompose selon (la démonstration est donnée en annexe)*[13] *:*

$$L(\pi) = L(\pi_{\exp}) + T \cdot l(\widetilde{s_i} \circ \pi^*, \pi; \pi_{\exp}) \quad [1]$$

où $\widetilde{s_i}$ est une distribution sur les états initiaux.

Ce lemme exprime le fait que la perte totale se décompose en une perte associée à la politique expert $\pi_{\exp}$ (la référence n'étant pas toujours atteignable dans l'espace de recherche) et en des pertes de potentiel par rapport à la politique expert associées à chaque action réalisée par la politique π. La perte associée à la politique expert est fixe et ne dépend pas de la politique π. Pour minimiser la perte totale, il suffit donc de minimiser la perte moyenne de potentiel par rapport à la politique expert :

$$\arg\min_{\pi \in \Pi} L(\pi) = \arg\min_{\pi \in \Pi} l(\widetilde{s_i} \circ \pi^*, \pi; \pi_{\exp}). \quad [2]$$

Autrement dit, il s'agit d'apprendre une politique qui prend les meilleures décisions locales possible comparativement à celles de l'expert. Le problème global d'apprentissage d'une structure est ainsi réduit à une suite de problèmes de classification. Nous étudions ci-dessous les difficultés de l'optimisation de [2], ainsi que les méthodes approximatives de la littérature.

2.2.2. *Ensemble d'apprentissage*

Le problème [2] est difficile à résoudre en raison de son aspect circulaire : sa résolution demande de minimiser une perte moyenne *sur une certaine distribution d'états*, qu'il importe de caractériser ; or, calculer cette distribution présuppose de connaître la politique utilisée pour construire des trajectoires le long desquelles ces états sont visités. Il est possible de rendre ce problème simple en sélectionnant une distribution fixe (c'est-à-dire indépendante de la politique) d'états $\widetilde{s}_{\text{fix}}$ pour remplacer la distribution $\widetilde{s_i} \circ \pi^*$. On pourra ensuite minimiser $l(\widetilde{s}_{\text{fix}}, \pi; \pi_{\exp})$ par rapport à la politique π. Avec un choix raisonnable pour la distribution $\widetilde{s}_{\text{fix}}$, on espère que cette fonction approxime correctement la fonction objectif $l(\widetilde{s_i} \circ \pi^*, \pi; \pi_{\exp})$.

Une fois que les états sont fixés, le problème d'optimisation devient un problème standard de classification multiclasse sensible aux coûts. Nous appelons les triplets (état, actions accessibles, pertes associées) les *situations* auxquelles une politique est confrontée ; ils représentent l'ensemble des informations à partir desquelles la politique doit faire le choix d'une action. En supposant que l'on sait calculer cet ensemble de situations, toute méthode permettant d'apprendre un classifieur sensible aux coûts permet de trouver la politique π minimisant l'objectif $l(\widetilde{s}_{\text{fix}}, \pi; \pi_{\exp})$[14].

La figure 2.(a) représente graphiquement un ensemble d'apprentissage composé de deux situations auxquelles la politique est confrontée dans les états 1 et 20. L'abscisse et l'ordonnée de chaque point-action correspondent aux deux caractéristiques

13. Techniquement, la boucle présente sur chaque état final nous permet de prolonger chaque chemin jusqu'à la longueur T même si l'état final est atteint plus tôt ; cela nous permet de consi-

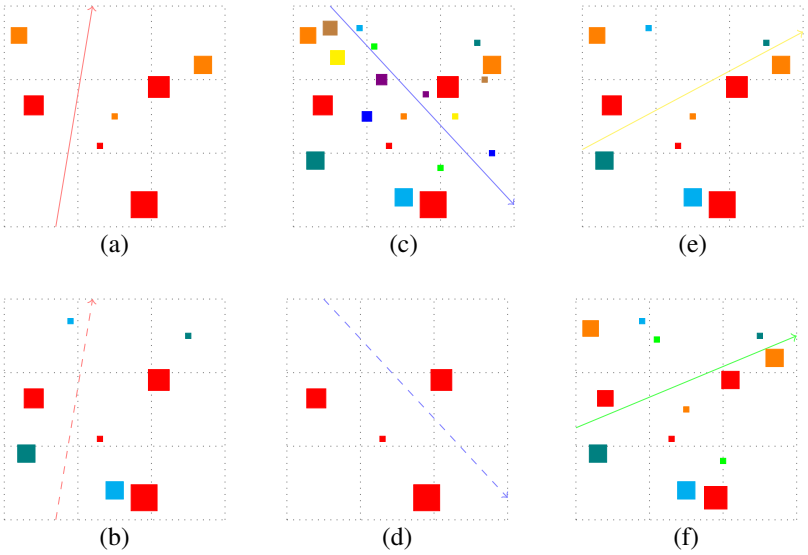

Figure 2. *Les ensembles d'apprentissage correspondant à l'application de différents algorithmes d'apprentissage à l'espace de recherche présenté sur la figure 1. La traduction de référence est « I am in a queue ». Chaque carré représente une action. L'abscisse et l'ordonnée de chaque point-action correspondent aux deux caractéristiques associées à cette action. Les couleurs correspondent aux couleurs des actions de la figure 1, tandis que les tailles des carrés correspondent (approximativement) à des pertes de potentiel associées. Parmi les points de même couleur, celle qui est la plus à gauche du séparateur correspond à l'action choisie. Les séparateurs continus correspondent à des classifieurs appris sur l'ensemble représenté, tandis que les séparateurs en pointillé sont donnés pour aider la visualisation et correspondent à des classifieurs qui doivent être appliqués sur les ensembles d'actions donnés.*

associées à cette action, et la taille des points représente (approximativement) la perte de potentiel[15] de l'action correspondant.

dérer uniquement les chemins de cette longueur. Pour des raisons de simplicité, nous utiliserons cette technique de manière systématique dans la suite.

14. Nous supposons dans cette section théorique que l'erreur liée à la taille limitée de l'ensemble d'apprentissage, ainsi qu'à l'estimation non exacte des coûts, est négligeable.

15. Si la politique par rapport à laquelle la perte de potentiel est calculée n'est pas indiquée on suppose par défaut que c'est la politique expert.

2.3. *Méthodes non itératives*

Nous commençons par étudier deux stratégies simples, dont dérivent deux instanciations possibles de $\widetilde{s}_{\text{fix}}$: (a) apprendre sur le chemin de l'expert (b) apprendre sur l'espace complet. L'étude de ces cas simples met clairement en évidence deux difficultés importantes qui se rencontrent également dans les approches plus complexes : l'*effet d'avalanche* et le *gaspillage de ressources*.

2.3.1. *Apprendre sur le chemin de l'expert : l'effet d'avalanche*

L'apprentissage sur le chemin de l'expert vise à imiter son comportement, en ne s'intéressant qu'aux situations qu'il rencontre, et donc à minimiser la perte suivante :

$$l(\widetilde{s}_i \circ \pi^*_{\text{exp}}, \pi; \pi_{\text{exp}}). \qquad [3]$$

Cette approche est problématique car les seules situations vues à l'apprentissage sont des situations « idéales ». Une fois la politique apprise, toute erreur lors de son application mène dans un état (non optimal), qui peut différer des états vus à l'apprentissage. Cette différence entre les états considérés à l'apprentissage et ceux qui sont rencontrés à l'inférence augmente le risque que de nouvelles erreurs soient commises, ce qui peut déclencher une réaction en chaîne. Apprendre sur les seules trajectoires expertes expose donc à *l'effet d'avalanche* [16]. La perte moyenne de potentiel [3] obtenue en apprentissage donne alors une mauvaise estimation de la perte totale $L(\pi)$.

Une borne possible pour la perte totale est la suivante (Ross *et al.*, 2011) :

$$L(\pi) \leq L(\pi_{\text{exp}}) + p_{\text{err}} \cdot T \cdot L_{\max}, \qquad [4]$$

où p_{err} est la probabilité de choisir une action différente de celle de l'expert dans un état *sur la trajectoire optimale*. Sur une trajectoire de longueur T, la probabilité de commettre une erreur est $p_{\text{err}} \cdot T$, et peut, au pire, conduire à une perte maximale $L_{\max}$, car même si une erreur n'implique qu'une perte de potentiel locale très petite, l'état d'arrivée de l'action choisie représente une situation inconnue pour la politique apprise, qui risque de conduire à de nouvelles erreurs. Cette garantie est insuffisante.

Les graphiques 2.(a) et 2.(b) illustrent cet effet d'avalanche. Comme le chemin de l'expert est $0 \rightarrow 1 \rightarrow 20 \rightarrow 21$, l'ensemble d'apprentissage sera composé des situations associées aux états 1 et 20 (dans les états 0 et 21, il n'y a pas de possibilité de choisir). Toutefois, avec une politique linéaire, les actions de l'expert ne peuvent pas être choisies, car ce chemin n'est pas linéairement séparable des autres. La politique apprise sur cet ensemble choisit alors les meilleures actions parmi les ensembles séparables. Le problème est que, en choisissant l'action $1 \rightarrow 3$, la politique doit réaliser des actions dans des états pour lesquels elle n'a pas été entraînée, et elle commet une erreur grave dans l'état 7, ce qui amène à une mauvaise traduction (0,39 BLEU).

16. Ce problème est aussi parfois appelé « *exposure bias* » (Ranzato *et al.*, 2016) ou encore « propagation d'erreur » (Lê et Fokkens, 2017) dans la littérature.

2.3.2. *Apprendre sur l'espace complet : le gaspillage de ressources*

Dans cette deuxième stratégie naïve, l'ensemble d'apprentissage comprend tous les états de l'espace de recherche. Soit $\widetilde{s}_{\text{uni}}$ la distribution uniforme sur l'ensemble des états. Supposons que l'on dispose d'un générateur de cette distribution, on peut alors, comme précédemment, résoudre un problème de classification sensible aux coûts pour trouver la politique qui minimise la perte de potentiel correspondante :

$$l(\widetilde{s}_{\text{uni}}, \pi; \pi_{\text{exp}}) \quad [5]$$

Inclure des exemples de toutes les situations pouvant se présenter à l'inférence soulève deux problèmes. D'abord un problème calculatoire, car l'espace de recherche pour une tâche structurée a généralement un nombre combinatoire d'états. Ensuite, même quand cette stratégie est techniquement réalisable, son efficacité n'est pas garantie, car la perte [5] ne reflète pas correctement la performance moyenne sur une trajectoire choisie par la politique π. Pour obtenir une garantie théorique, une possibilité est d'utiliser sa borne supérieure $\sup_{s\sim\widetilde{s}_{\text{uni}}}[l(s, \pi; \pi_{\text{exp}}]$ selon :

$$L(\pi) \leq L(\pi_{\text{exp}}) + \sup_{s\sim\widetilde{s}_{\text{uni}}}[l(s, \pi; \pi_{\text{exp}}] \cdot T \quad [6]$$

Intuitivement, le deuxième problème est lié au fait que, dans un cadre d'apprentissage automatique tel que le nôtre, les ressources du système sont limitées. Si tous les états possibles sont considérés lors de l'apprentissage, une partie des paramètres du classifieur sera consacrée à modéliser des phénomènes très éloignés du chemin suivi par l'expert et qui ne seront que très peu utiles lors de l'inférence.

Les figures 2.(c) et 2.(d) illustrent le gaspillage de ressources. L'ensemble de situations auxquelles la politique peut être confrontée est linéairement non séparable, le meilleur compromis étant le séparateur bleu figure 2.(c). Parmi les neuf situations « apprises », la politique associée à ce séparateur n'en rencontre qu'une seule pendant l'inférence. On voit figure 2.(d) que le séparateur choisi n'est pas optimal pour cette situation et amène à une mauvaise traduction (0,36 BLEU).

3. Apprendre à chercher : itération de politique

3.1. *Principe général*

La section précédente a montré que la composition de l'ensemble d'apprentissage de la politique a une grande importance : il est inutile de gaspiller des ressources en apprenant des situations que l'on ne rencontrera pas en pratique, tout comme il est risqué de se retrouver dans une situation que l'on n'a pas étudiée pendant l'apprentissage.

La notion *d'itération de politique* est à la base de toutes les méthodes d'apprentissage présentées ci-dessous. L'idée générale de ces méthodes, parfois également appelées *interactives* (Sun *et al.*, 2017), est d'apprendre une série de politiques, en utilisant les politiques déjà apprises pour construire un nouvel ensemble d'apprentissage et apprendre une nouvelle politique.

Utiliser une ou plusieurs politiques apprises antérieurement pour construire les nouvelles instances d'apprentissage permet de créer un ensemble d'états proches des conditions de l'inférence. Dans certaines situations, il est préférable d'utiliser également une politique apprise pour évaluer les coûts des actions. Cela fournira des coûts plus réalistes, car les pertes locales « expertes » sont fondées sur des anticipations souvent trop optimistes du futur (Chang *et al.*, 2015b). Deux politiques déterminent donc l'ensemble d'apprentissage : π_{in} la politique d'entrée (*roll-in policy*), et π_{out} est la politique de sortie (*roll-out*). On atteint les états qui constituent l'ensemble d'apprentissage en utilisant π_{in} ; on évalue la perte locale associée aux actions en supposant que la suite du calcul utilise π_{out}. On note l'ensemble d'apprentissage obtenu $\mathrm{Clf\_Prob}(\pi_{\mathrm{in}}, \pi_{\mathrm{out}})$. Ces bases posées, nous présentons ci-dessous deux familles d'algorithmes « apprendre à chercher », DAGGER et SEARN.

3.2. *La famille DAGGER*

Dans les algorithmes de la famille DAGGER[17], la politique est apprise sur l'accumulation de tous les exemples collectés depuis le début de la procédure d'entraînement. La philosophie de cette approche est d'apprendre à agir dans de nouvelles situations tout en préservant les connaissances accumulées lors des itérations précédentes. Asymptotiquement, cela permet de trouver une politique adaptée à l'ensemble des situations qu'elle peut potentiellement rencontrer. En revanche, l'ensemble d'apprentissage risque de contenir plus de situations que celles qui sont utiles en pratique, ce qui expose au problème du gaspillage de ressources.

3.2.1. *L'algorithme AggreVaTe*

L'algorithme AggreVate (Ross et Bagnell, 2014) part de la politique expert, et à chaque itération construit un nouvel ensemble d'apprentissage $\mathrm{Clf\_Prob}(\pi_{n-1}, \pi_{\mathrm{exp}})$ en utilisant la politique de l'itération précédente, les pertes associées aux actions étant évaluées par rapport à la politique expert. Cet ensemble est fusionné avec celui de l'itération précédente et une nouvelle politique est apprise en résolvant le problème de classification ainsi posé. La politique finale est choisie à l'aide de données de validation parmi les politiques apprises. Cette procédure correspond à l'algorithme 1[18].

Pour la classe de politiques convexes et en supposant un nombre infini d'itérations, on montre que les performances de la politique ainsi obtenue sont égales aux performances de la *meilleure* politique pour un ensemble de trajectoires accumulées au cours de l'apprentissage. Ce résultat est formalisé dans le théorème 3.1.

17. DAGGER (Ross *et al.*, 2011) est historiquement l'algorithme le plus connu de cette famille. Néanmoins, nous fondons notre présentation sur un algorithme plus récent, AggreVaTe (Ross et Bagnell, 2014) qui possède de meilleures garanties théoriques.

18. Dans cet article nous présentons une version légèrement simplifiée de AggreVaTe (qui permet toutefois d'obtenir les mêmes garanties théoriques) ; une comparaison avec la version originale se trouve en annexe.

Algorithme 1 : AggreVaTe : agrégation des données d'apprentissage

Données : données d'apprentissage et de validation $\sim \widetilde{D}$, ensemble paramétrique Π

Initialisation : $\pi_0 \leftarrow$ politique arbitraire de Π, $\mathcal{E} \leftarrow \emptyset$;
pour $n = 1..N$ **faire**
    Construire le problème d'apprentissage $\mathcal{E}_n = \text{Clf\_Prob}(\pi_{n-1}, \pi_{\text{exp}})$;
    Fusionner les exemples d'apprentissage $\mathcal{E} = \mathcal{E} \cup \mathcal{E}_n$;
    Apprendre π_n en minimisant la perte sur $\mathcal{E}$;
fin
En utilisant les données de validation, choisir π_{res} parmi $\{\pi_n\}_{n=1}^N$;
retourner π_{res};

Théorème 3.1 *Soit* Π *un ensemble paramétrique convexe de politiques,* $\breve{l}$ *un majorant convexe de la perte de potentiel*[19] *par rapport aux paramètres définissant* Π*. AggreVaTe construit une suite* $\{\pi_n\}_{n=1}^N$ *telle que pour au moins une politique* π_{res} *on a :*

$$L(\pi_{\text{res}}) \leq L_{\text{exp}} + T \cdot \min_{\pi \in \Pi} \frac{1}{N} \sum_{n=1}^{N} \breve{l}(\widetilde{s_i} \circ \pi_n, \pi, \pi_{\text{exp}}) + O\left(\frac{1}{N}\right) \qquad [7]$$

La meilleure politique π_{res} *peut être sélectionnée à l'aide de l'ensemble de validation.*

Notons que la garantie obtenue n'est pas une garantie d'optimalité. Minimiser la perte $\sum_{n=1}^{N} \breve{l}(\widetilde{s_i} \circ \pi_n^*, \pi, \pi_{\text{exp}})$ sur les trajectoires obtenues avec un ensemble de politiques plutôt que sur les seuls états obtenus avec la politique courante $\breve{l}(\widetilde{s_i} \circ \pi^*, \pi, \pi_{\text{exp}})$ soulève le problème du gaspillage de ressources. En fait, rien ne garantit ni la qualité de l'ensemble de trajectoires considéré $\cup_{n=1}^{N} \widetilde{s_i} \circ \pi_n^*$ ni la bonne allocation des ressources d'apprentissage entre des trajectoires plus ou moins pertinentes.

Un exemple de déroulement de AggreVaTe est donné par les figures 2.(a) et 2.(e). Comme nous l'avons vu, le séparateur rouge de la figure 2.(a) a été choisi en prenant en compte seulement les états 1 et 20. À l'inférence, d'autres états sont rencontrés et ajoutés dans l'ensemble d'apprentissage. Le nouveau séparateur jaune (figure 2.(e)) trouve la deuxième meilleure traduction de l'espace de recherche avec le score 0,63. L'algorithme a convergé[20].

19. La démonstration de ce théorème repose sur la théorie de l'optimisation convexe en ligne où la fonction de perte est supposée être convexe. De manière générale, ce n'est pas le cas pour la perte de potentiel définie dans la section 2.1.4, c'est pourquoi un majorant convexe est utilisé ici (Ross et Bagnell, 2014). Notons que cela implique également une limitation sur la classe de politiques considérée qui est restreinte aux politiques convexes.

20. La convergence rapide est ici due à l'heureuse disposition des points sur le plan, un choix fait par simplicité. Dans le cas général, la convergence n'est pas garantie et la meilleure politique doit en général être choisie à l'aide d'un ensemble de validation (voir l'annexe pour plus de détails).

3.3. *La famille SEARN*

L'idée générale de SEARN (Daumé *et al.*, 2009) est d'apprendre progressivement une politique en partant de la politique expert et en confiant de plus en plus de décisions à la politique apprise, jusqu'à ce que cette politique ne fasse plus appel à l'expert. Dans cet article, nous présentons SEARN de manière légèrement différente de la version originale (Daumé *et al.*, 2009), ce qui simplifie les garanties théoriques associées. Les différences avec la version originale sont discutées en annexe.

Algorithme 2 : SEARN

Données : données d'apprentissage $\sim \widetilde{D}$, ensemble paramétrique Π, nombre d'itérations N, politique expert déterministe π_{exp}

Initialisation : $\pi_0 \leftarrow \pi_{\text{exp}}$;
pour $n \in 1..N$ **faire**
  Construire l'ensemble d'apprentissage $\mathcal{E}_n = \text{Clf\_Prob}(\pi_{n-1}, \pi_{n-1})$;
  Apprendre τ_n avec les exemples $\mathcal{E}_n$;
  $\pi_n = \text{Interpol}(\pi_{n-1}, \tau_n)$;
fin
retourner $\bar{\pi}_N$;

SEARN utilise la politique $\pi_n = \text{Interpol}(\pi_{n-1}, \tau_n)$ qui est une politique composite. Elle utilise le classifieur τ_n pour prendre une unique décision (aléatoire) sur une trajectoire et utilise la politique π_{n-1} pour les autres décisions. Ce qui donne pour π_n la formule récursive suivante :

$$\pi_n = \text{Interpol}(\pi_{n-1}, \tau_n) = \begin{cases} \tau_n & \text{pour } \hat{t} = \text{rand}(1, T) \\ \pi_{n-1} & \text{pour } t \neq \hat{t} \end{cases} \quad [8]$$

Pour formuler les garanties théoriques, nous introduisons une nouvelle notion de perte, calculée par rapport à l'action choisie par une autre politique. Nous avons besoin de comparer le cas où le chemin est obtenu en utilisant la politique π_1 à l'exception d'une position où la politique π_2 est utilisée, par rapport au cas où π_1 construit le chemin entier. Cela représente une perte liée à l'utilisation locale de π_2 :

$$l_{\text{loc}}(\pi_2; \pi_1) = V(\widetilde{s}_i; \pi_1) - Q(\widetilde{s}_i \circ \pi_1^*, \pi_2; \pi_1).$$

Les garanties théoriques de SEARN sont alors données par le théorème suivant :

Théorème 3.2 *Soit $\bar{\pi}_N$ la politique obtenue après N itérations de SEARN simple et après le remplacement des appels à l'expert par des décisions aléatoires. L'inégalité suivante est alors vérifiée*[21] *:*

$$L(\bar{\pi}_N) \leq L(\pi_{\text{exp}}) + \sum_{n=1}^{N} l_{\text{loc}}(\tau_n; \pi_{n-1}) + T \left(\frac{T-1}{T} \right)^N L_{\max} \quad [9]$$

21. Voir la démonstration complète en annexe.

où $L_{\max}$ (constante) est la perte maximale possible dans les espaces de recherche $\tilde{D}$.

Ce théorème signifie que la qualité de chaque nouvelle politique apprise est globalement déterminée par la somme des pertes *locales* obtenues à chaque itération. Les changements lents et contrôlés de la politique permettent d'assurer que l'effet d'avalanche est quasiment absent. Après N itérations, il existe encore une faible probabilité que la politique π_N fasse appel à l'expert, qui n'est pas disponible à l'inférence. On suppose que dans cette situation, la décision est prise aléatoirement, et implique la pire perte possible $L_{\max}$. Mais comme la probabilité de faire appel à l'expert diminue chaque itération, cela ne rajoute finalement qu'un petit terme supplémentaire dans la perte associée à la politique finale. Pour fixer les ordres de grandeur, $(T-1)^N T^{-(N-1)} \approx 2 \times 10^{-4}$ pour $T = 10$ et $N = 100$.

Un exemple de déroulement de SEARN est donné par les figures 2.(a) et 2.(f). Le séparateur rouge trouvé à la première itération (2.(a)) ne tient pas compte du fait que l'action $1 \rightarrow 3$, choisie dans l'état 1, amènera dans les états non explorés lors de l'apprentissage. Lors de la deuxième itération les états 3 et 7 rencontrés après avoir effectué cette action sont ajoutés dans l'ensemble d'apprentissage (ainsi que l'état 4 qui est à la fin d'un autre chemin exploré par SEARN)[22]. Le chemin $0 \rightarrow 1 \rightarrow 3 \rightarrow 7 \rightarrow 12$ choisi par le nouveau séparateur vert (2.(f)) est séparable et de bonne qualité ; cela termine l'algorithme. Notons que les pertes associées aux actions rouges sont changées, car ce ne sont plus les pertes de potentiel vis-à-vis de la politique expert, mais vis-à-vis de la politique composite qui choisit une action sur 6 (6 étant la longueur maximale du chemin) à l'aide de la politique « rouge » . Notamment, les pertes associées aux actions $1 \rightarrow 3$, $1 \rightarrow 2$, $1 \rightarrow 5$ sont diminuées par rapport à la perte de l'action de l'expert $1 \rightarrow 20$, car la politique composite a une probabilité non nulle de faire un mauvais choix dans l'état 20.

3.3.1. *Comparaison avec AggreVaTe*

Un point important concernant les garanties de SEARN est qu'elles ne donnent aucune information sur la qualité de la politique obtenue par rapport aux autres politiques de la classe considérée. On ne sait notamment pas si elle est proche de la politique optimale de cette classe ou non. Ces garanties ne nous informent que sur l'écart par rapport à l'expert. C'est une différence importante avec AggreVaTe qui garantit d'obtenir une politique presque optimale pour un certain nombre de problèmes, mais qui n'évite toutefois pas le gaspillage de ressources.

Notons également que, contrairement à AggreVaTe, pour obtenir les garanties [9], SEARN a besoin aussi bien des scores d'expert que des scores du tuteur, ce qui complique son utilisation en pratique. En revanche, SEARN ne met pas de limitation sur

22. Comme SEARN est un algorithme stochastique, pour donner une illustration précise, il faudrait ajouter les probabilités d'apparition de chaque situation dans l'ensemble d'apprentissage. Ici nous ne prenons pas en compte ce détail pour ne pas alourdir l'illustration, car dans tous les cas le chemin choisi par le séparateur sera le même (parce que ce chemin est séparable).

la classe de politiques considérée, alors que les garanties théoriques de AggreVate ne sont valables que pour les politiques convexes.

3.4. *Références supplémentaires*

Les sections précédentes nous ont confrontés au problème de la poule et de l'œuf : pour entraîner la politique optimale, il faut construire un bon problème d'apprentissage, mais pour construire celui-ci, la politique doit déjà être connue. Les méthodes L2S se ramènent alors à un problème d'optimisation de la répartition des ressources afin d'obtenir la politique optimale, contrairement à *l'apprentissage par renforcement* ou *par imitation*, qui font face à des difficultés supplémentaires d'exploration.

Le vocabulaire utilisé dans cet article (politique, fonction de valeur, récompense...) est en fait largement emprunté à l'apprentissage par renforcement (Sutton et Barto, 1998), tandis que l'utilisation que nous faisons de l'expert provient de l'apprentissage par imitation ; néanmoins, la nature du problème considéré ici est relativement différente. Dans le cadre L2S, nous disposons d'un accès direct aux pertes de potentiel pour chaque action, contrairement à l'apprentissage par renforcement où il est en général possible de n'observer que des informations partielles sur la récompense cumulée finale et où l'accès au choix de l'expert n'est jamais envisagé. De même, la disponibilité d'un simulateur puissant permettant d'obtenir les pertes de potentiel nous éloigne également de l'apprentissage par imitation où généralement seules des traces du comportement de l'expert sont disponibles (Geist, 2016).

De nombreuses stratégies présentées dans cette section sont en fait bien connues dans le cadre de l'apprentissage par renforcement. Par exemple, le problème de gaspillage de ressources se retrouve dans le théorème sur la qualité d'une politique gloutonne fondée sur la fonction de valeur approximée dans (Bertsekas, 1987, p. 236). De même, le principe itératif introduit dans la section 3.1 ressemble au concept d'itération de politique approximée (*approximated policy iteration*, (Bertsekas et Tsitsiklis, 1996)). Finalement, Kakade (2003) introduit l'algorithme d'itération conservative de politique (*Conservative Policy Iteration*), élaboré pour le cas de l'horizon infini et également sans appel à l'expert, qui est très similaire à SEARN.

4. « Apprendre à chercher » : applications au TAL

4.1. *Des applications emblématiques et les espaces de recherche associés*

Les diverses structures linguistiques présentes dans la langue naturelle fournissent un cadre d'application intéressant pour la prédiction structurée et donc pour les méthodes « apprendre à chercher ». Un premier cas de structure très simple est celui de la séquence de symboles, utilisée dans des tâches telles que la reconnaissance de l'écriture manuscrite, la reconnaissance d'entités nommées, l'étiquetage morpho-syntaxique ou le calcul de la prononciation d'une chaîne orthographique. Dans toutes

ces tâches, l'objectif est de mettre en correspondance chaque élément d'une séquence « source » avec un élément d'une séquence cible. Il s'agit d'un cadre très simple d'application pour les méthodes qui nous intéressent, étudié par exemple dans (Daumé *et al.*, 2009 ; Ross *et al.*, 2011 ; Doppa *et al.*, 2014 ; Knyazeva *et al.*, 2015). Pour ces applications, la procédure standard de recherche consiste à effectuer des actions attribuant une étiquette aux différentes positions l'une après l'autre, en utilisant la séquence source et les étiquettes déjà connues pour composer l'état, ce qui permet de mieux informer les actions ultérieures.

La tâche d'analyse en dépendances possède un espace de sorties composé des arbres syntaxiques valides pour une phrase. Dans le cas des analyseurs par transition (Nivre, 2008), la recherche de l'arbre optimal peut être effectuée en réalisant des actions telles que *shift*, *reduce* ou autres[23](Goldberg et Nivre, 2012 ; Chang *et al.*, 2015a ; Chang *et al.*, 2015b). Comme il est typique pour les applications L2S en TAL, l'état est composé de la phrase source ainsi que des informations de toutes les actions déjà effectuées.

La tâche d'extraction d'évènements biomédicaux est similaire : elle vise à fournir pour chaque phrase un graphe dirigé sans cycle, dans lequel les relations entre les mots composant les évènements de différents types sont établies. Vlachos et Craven (2011) l'abordent à l'aide des techniques L2S : lors de l'inférence, plusieurs passes sur la phrase sont effectuées avec comme objectifs successifs de détecter les évènements, leurs arguments et enfin d'établir la structure complète. Chaque sous-tâche pourrait donner lieu à un problème structuré distinct ; la modélisation L2S permet de les intégrer au sein d'une unique recherche, une approche qui est hors de portée des techniques classiques, au vu de la complexité de l'espace de recherche.

La tâche de résolution de coréférences a pour objectif de détecter les formes qui se réfèrent à un même objet « réel » en calculant les regroupements correspondants. Dans (Clark et Manning, 2016), ces regroupements sont construits en réunissant progressivement les groupes de mots en partant des mots isolés et en finissant par un groupe pour chaque référent. Cette approche se prête bien au cadre L2S, avec des actions qui correspondent aux décisions de fusion de groupes de mentions, et un ensemble d'états correspondant aux regroupements partiellement construits.

4.2. *Quelques détails d'implantation*

Les applications d'« apprendre à chercher » au TAL se caractérisent par une certaine liberté dans la construction de l'algorithme, ne correspondant pas toujours au cadre théorique précis. Par exemple, les mises à jour de la politique peuvent être fréquentes plutôt qu'à la fin de chaque époque (Goldberg et Nivre, 2012), le mélange stochastique des politiques dans les méthodes de la famille SEARN peut être construit de manière approximative (Daumé *et al.*, 2009), la contrainte de convexité de la politique

23. L'ensemble des actions possibles dépend de l'architecture de l'analyseur syntaxique.

pour les méthodes de la famille AggraVaTe n'est pas nécessairement remplie (Chang *et al.*, 2015a), etc. Ces applications ont toutefois en commun les grands principes des méthodes L2S, tels que :

– le principe itératif ;

– l'utilisation de la politique expert (*reference roll-in*) ainsi que de la politique apprise (*learned roll-in*) afin d'explorer les états pertinents de l'espace de recherche ;

– l'estimation soigneuse des scores des actions en prenant en compte non seulement la récompense immédiate apportée par l'action, mais aussi son impact global sur la sortie complète (évaluation par l'expert, ou *reference roll-out*) et dans certains cas les imperfections de la politique poursuivant l'action (évaluation par le tuteur, ou *learned roll-out*) ;

– un changement lent de la politique d'une itération à la suivante (ce qui peut être fait par mélange stochastique comme dans SEARN ou par aggrégation des ensembles d'apprentissage comme dans AggreVaTe, ainsi qu'en utilisant les mises à jour en ligne comme proposé dans (Goldberg et Nivre, 2012)).

4.2.1. *Le calcul de l'expert*

Comme expliqué section 2.1.4, les méthodes « apprendre à chercher » utilisent la politique expert pour déterminer l'action optimale à partir d'un état arbitraire de l'espace de recherche (en supposant que les actions suivantes sont elles aussi choisies de manière optimale). En TAL, cette politique peut être plus ou moins simple à calculer selon la structure de l'espace de recherche et de la métrique d'évaluation.

Les tâches d'étiquetage de séquences évaluées en utilisant le score de Hamming disposent d'un expert trivial dans tous les états de l'espace de recherche : c'est l'action proposant l'étiquette de référence pour la position considérée. Dans le cas du F-score, le calcul est également direct (Daumé *et al.*, 2009). Pour certains systèmes d'analyse en dépendances, une méthode de calcul de l'expert en temps constant connue sous le nom « d'oracle dynamique » a été également proposée dans (Goldberg et Nivre, 2012).

Pour d'autres tâches, en revanche, l'expert n'est pas calculable en temps raisonnable. C'est le cas, par exemple, de la résolution de coréférences, où la métrique d'évaluation utilisée est complexe et ne se décompose pas sur les actions isolées (Ma *et al.*, 2014). Cette propriété de non-décomposabilité empêche l'utilisation de programmation dynamique pour calculer efficacement la politique expert. Dans de telles situations, une approximation de la politique expert peut être utilisée, par exemple la politique choisissant l'action apportant l'amélioration maximale immédiate (Clark et Manning, 2016). Une autre approximation possible consiste à utiliser toujours l'étiquette de référence, même si, en combinaison avec les décisions passées, elle n'est pas nécessairement optimale (Venkatraman *et al.*, 2015).

4.2.2. *Le calcul du tuteur*

Le calcul du score du tuteur présente des difficultés différentes de celui de l'expert. D'une part, la politique future étant fixée, l'évaluation concerne donc uniquement

l'action en cours, contrairement au cas de la politique expert qui prend sa décision en prévoyant aussi bien l'action en cours que toutes les autres actions restant à réaliser. D'autre part, il est nécessaire de recalculer ces scores pour chaque nouvelle politique future considérée. Pour les applications au TAL, le score du tuteur est en général calculé en effectuant une simulation reposant sur l'utilisation de la politique dont on souhaite calculer le score (Daumé *et al.*, 2009). Dans la mesure où cette politique peut être stochatique, afin de réduire la variance, cette estimation peut être réalisée à l'aide de plusieurs simulations. Cette reproductibilité est assez particulière au domaine de TAL, elle est due d'une part à la longueur relativement petite des séquences d'actions et d'autre part à l'absence des facteurs aléatoires de l'environnement (ce qui n'est pas le cas par exemple dans la robotique). En revanche, l'utilisation de plusieurs simulations augmente le coût de l'algorithme.

Une approximation souvent employée est l'utilisation de la politique expert à la place de la politique par rapport à laquelle le score doit être calculé (si l'expert est facilement calculable). En pratique cette approximation donne souvent de bons résultats, spécialement si la politique apprise est de qualité proche de l'expert (Daumé *et al.*, 2009 ; Clark et Manning, 2016). Néanmoins, pour certaines tâches le calcul soigneux des scores vis-à-vis de la politique courante se montre important. C'est le cas par exemple pour l'extraction d'évènements biomédicaux (Vlachos et Craven, 2011), ainsi que pour l'analyse en dépendances (Chang *et al.*, 2015b) (pour ce dernier, l'avantage d'utiliser les scores du tuteur, et plus précisément un mélange des scores de l'expert et du tuteur, a été établi pour le cas où la référence est sous-optimale).

4.3. *Motiver l'utilisation de méthodes L2S en TAL*

Pour conclure cette section, notons qu'en TAL plusieurs raisons peuvent motiver l'utilisation de l'inférence approchée par rapport aux méthodes classiques fondées sur la programmation dynamique, ce qui à son tour peut motiver l'utilisation des méthodes « apprendre à chercher »[24].Tout d'abord, la programmation dynamique peut être appliquée uniquement pour les structures relativement simples, comme des séquences et des arbres. Certaines tâches, exemplifiées ici par l'extraction d'évènements biomédicaux, emploient des structures plus complexes, qui sont en dehors du champ d'application de la programmation dynamique (Vlachos et Craven, 2011). On peut de même vouloir utiliser pour l'apprentissage des fonctions de perte non décomposables sur les arcs de l'espace de recherche (comme BLEU (Papineni *et al.*, 2002) en traduction automatique), ce qui reste faisable dans un cadre L2S. Enfin, même si pour certaines tâches, la programmation dynamique permet de réaliser l'apprentissage et l'inférence avec une complexité polynomiale, en pratique cette complexité peut rester prohibitive : c'est le cas de tâches d'étiquetage qui manipulent de très grands ensembles d'étiquettes (Müller *et al.*, 2013 ; Lavergne et Yvon, 2017).

24. En pratique l'utilisation des techniques L2S permet souvent d'améliorer le résultat si la recherche utilisée est gloutonne ou en faisceau, mais ce n'est pas toujours le cas : certains travaux rapportent l'absence d'améliorations perceptibles (Wiseman *et al.*, 2016).

En dehors des problèmes de complexité, l'utilisation de l'inférence approchée peut être motivée par la richesse des informations contenues dans les dépendances longues. Ainsi, il peut être bénéfique de sacrifier l'exactitude de la recherche afin de permettre l'utilisation de dépendances à longue distance, c'est le cas, par exemple, pour l'analyse en dépendances (Zhang et Nivre, 2011 ; Goldberg et Nivre, 2012), l'analyse syntaxique (Collins et Roark, 2004) ou encore des diverses tâches d'étiquetage de séquences (Daumé *et al.*, 2009). Cet aspect peut notamment être renforcé par la modification de la procédure de recherche avec pour objectif de permettre un ordre libre pour la prise de décisions. Ainsi, les décisions complexes (attribution d'une étiquette morphosyntaxique pour un mot ambigu, reconnaissance d'une lettre non clairement écrite) peuvent être retardées et réalisées en utilisant le maximum d'informations sur l'ensemble de la structure (Shen *et al.*, 2007 ; Knyazeva *et al.*, 2015).

Une autre limitation importante imposée par la programmation dynamique est qu'elle restreint fortement la famille de classifieurs pouvant être utilisée, en pratique seuls des classifieurs linéaires semblent être utilisés. Cette limitation peut être en partie contournée en ajoutant des caractéristiques adaptées (manuellement ou bien encore à l'aide de noyaux polynomiaux). Cette « astuce » ne permet toutefois pas d'utiliser des classifieurs plus expressifs tels que les réseaux de neurones récurrents qui restent incompatibles avec les méthodes de programmation dynamique.

5. Conclusion

Dans cet article, nous avons présenté un panorama d'une famille de méthodes permettant de réaliser des tâches d'apprentissage structuré, qui sont au cœur des méthodes du TAL moderne. Contrairement aux méthodes standard d'apprentissage structuré, qui reposent souvent sur des simplifications du modèle de dépendances, les méthodes de la famille L2S s'appuient sur des simplifications de l'algorithme de recherche, supposé effectuer un décodage glouton, ce qui permet d'une part de prendre en compte des dépendances plus complexes, d'autre part de considérer des fonctions de pertes plus riches. Après avoir présenté les bases théoriques qui sous-tendent ces méthodes et illustré leur fonctionnement sur un problème de traduction automatique, nous avons discuté leur utilisation sur diverses tâches du TAL.

Parmi les questions encore ouvertes, mentionnons celles qui s'intéressent à l'utilisation d'autres formes (simples) pour réaliser l'inférence. Il est ainsi connu qu'une manière simple pour améliorer la recherche gloutonne consiste à employer des techniques de recherche en faisceau (*beam search*). La généralisation des méthodes L2S à la recherche en faisceau est difficile, car cette technique considère simultanément plusieurs états de l'espace de recherche : définir dans ce contexte l'équivalent de l'effet de gaspillage de ressources et de l'effet d'avalanche n'est pas immédiat et plusieurs travaux s'inscrivent dans cette lignée pour des tâches d'apprentissage structuré en TAL (Collins et Roark, 2004 ; Huang *et al.*, 2012 ; Aufrant *et al.*, 2017).

Mais l'actualité principale de ces méthodes est liée à l'utilisation de réseaux de neurones récurrents, qui constituent l'état de l'art pour de nombreuses tâches du TAL, comme l'analyse en dépendances (Dyer *et al.*, 2015) et la traduction automatique (Bahdanau *et al.*, 2015). La forte analogie avec le cadre L2S – les décisions sont prises les unes après les autres par « les unités » du réseau récurrent (GRU ou LSTM) jouant un rôle similaire à celui du classifieur local –, et l'impossibilité d'utiliser des méthodes de recherche exacte, font que les concepts et techniques L2S trouvent une claire utilité dans ce cadre. Ainsi, Bengio *et al.* (2015) obtiennent des gains importants pour la tâche de traduction grâce à l'utilisation pendant l'entraînement de la politique apprise ; ces prédictions sont introduites progressivement au cours des itérations, comme dans le cas de SEARN. Ballesteros *et al.* (2016) poursuivent la ligne générale du travail initié par Goldberg et Nivre (2012) en utilisant l'oracle dynamique dans les RNN pour l'analyse en dépendances. Wiseman et Rush (2016) ont adapté une forme plus ancienne d'« apprendre à chercher », LASO (Daumé III et Marcu, 2005), pour l'entraînement des RNN avec une recherche en faisceau et obtenu des succès à la fois en traduction et en analyse en dépendances. Une adaptation d'AggreVaTe a enfin été proposée dans (Sun *et al.*, 2017) avec une modification de l'algorithme d'optimisation sans regret. Ses performances ont été validées sur la tâche d'analyse en dépendances d'équations manuscrites. Dans ces applications, la transposition directe des garanties théoriques des méthodes L2S n'est pas immédiate, comme discuté récemment par Leblond *et al.* (2018) dans leur extension de SEARN aux cas des RNN.

Des questions théoriques et pratiques bien étudiées dans le formalisme L2S se posent de manière proche dans des formalismes apparentés (l'apprentissage par renforcement) comme dans diverses applications utilisant des réseaux récurrents. Nous espérons que la présentation synthétique qui a été donnée ici de ce formalisme plus simple sera utile pour comprendre ces problèmes et leur apporter des réponses idoines.

6. Bibliographie

Aufrant L., Wisniewski G., Yvon F., « Don't Stop Me Now ! Using Global Dynamic Oracles to Correct Training Biases of Transition-Based Dependency Parsers », *Proc. EACL*, Valencia, Spain, p. 318-323, 2017.

Bahdanau D., Cho K., Bengio Y., « Neural machine translation by jointly learning to align and translate », *Proc. ICLR*, 2015.

Bakir G. H., Hofmann T., Schölkopf B., Smola A. J., Taskar B., Vishwanathan S. V. N., *Predicting Structured Data*, MIT Press, 2007.

Ballesteros M., Goldberg Y., Dyer C., Smith N. A., « Training with Exploration Improves a Greedy Stack LSTM Parser », *Proc. EMNLP*, Austin, TX, p. 2005-2010, November, 2016.

Bengio S., Vinyals O., Jaitly N., Shazeer N., « Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks », *Proc. NIPS*, Montreal, Canada, p. 1171-1179, 2015.

Bertsekas D. P., *Dynamic Programming : Deterministic and Stochastic Models*, Prentice-Hall, Inc., 1987.

Bertsekas D. P., Tsitsiklis J. N., *Neuro-Dynamic Programming*, 1st edn, Athena Scientific, 1996.

Chang K., He H., III H. D., Langford J., « Learning to Search for Dependencies », *arXiv preprint arXiv :1503.05615*, 2015a.

Chang K.-W., Krishnamurthy A., Agarwal A., Daumé III H., Langford J., « Learning to search better than your teacher », *Proc. ICML*, Lille, France, p. 2058-2066, 2015b.

Clark K., Manning C. D., « Improving Coreference Resolution by Learning Entity-Level Distributed Representations », *Proc. ACL*, Berlin, Germany, p. 643-653, 2016.

Collins M., Roark B., « Incremental Parsing with the Perceptron Algorithm », *Proc. ACL*, Barcelona, Spain, p. 111-118, July, 2004.

Daumé III H., Langford J., Marcu D., « Search-based Structured Prediction », *Machine Learning*, vol. 75, nº 3, p. 297-325, June, 2009.

Daumé III H., Marcu D., « Learning as Search Optimization : Approximate Large Margin Methods for Structured Prediction », *Proc. ICML*, Bonn, Germany, p. 169-176, 2005.

Doppa J. R., Fern A., Tadepalli P., « HC-Search : A Learning Framework for Search-based Structured Prediction », *JAIR*, vol. 50, p. 369-407, 2014.

Dyer C., Ballesteros M., Ling W., Matthews A., Smith N. A., « Transition-Based Dependency Parsing with Stack LSTM », *Proc. ACL-IJCNLP*, Beijing, China, p. 334-343, 2015.

Elkan C., « The Foundations of Cost-sensitive Learning », *Proceedings of IJCAI 2001*, San Francisco, CA, USA, p. 973-978, 2001.

Geist M., Contrôle optimal et apprentissage automatique, applications aux interactions homme-machine, Habilitation à diriger des recherches, Université de Lille 1, Février, 2016.

Goldberg Y., Nivre J., « A Dynamic Oracle for Arc-Eager Dependency Parsing », *Proceedings of COLING 2012*, Mumbai, India, p. 959-976, December, 2012.

Huang L., Fayong S., Guo Y., « Structured Perceptron with Inexact Search », *Proc. ACL-HLT*, Montréal, Canada, p. 142-151, June, 2012.

Kakade S., On the Sample Complexity of Reinforcement Learning, PhD thesis, Gatsby Computational Neuroscience Unit, University College London, 2003.

Knyazeva E., Wisniewski G., Yvon F., « Apprentissage par imitation pour l'étiquetage de séquences : vers une formalisation des méthodes d'étiquetage "easy-first" », *Proc. TALN*, Caen, France, p. (12), 2015.

Lavergne T., Yvon F., « Learning the Structure of Variable-Order CRFs : a finite-state perspective », *Proc. EMNLP*, Copenhagen, Denmark, p. 433-439, 2017.

Leblond R., Alayrac J.-B., Osokin A., Lacoste-Julien S., « SEARNN : Training RNNs with global-local losses », *Proc. ICLR*, 2018.

Liang P., Bouchard-Côté A., Klein D., Taskar B., « An End-to-end Discriminative Approach to Machine Translation », *Proc. COLING-ACL*, Sydney, Australia, p. 761-768, 2006.

Lê M., Fokkens A., « Tackling Error Propagation through Reinforcement Learning : A Case of Greedy Dependency Parsing », *Proc. EACL*, Valencia, Spain, p. 677-687, 2017.

Ma C., Doppa J. R., Orr J. W., Mannem P., Fern X. Z., Dietterich T. G., Tadepalli P., « Prune-and-Score : Learning for Greedy Coreference Resolution », *Proc. EMNLP*, Doha, Qatar, p. 2115-2126, 2014.

Müller T., Schmid H., Schütze H., « Efficient Higher-Order CRFs for Morphological Tagging », *Proc. EMNLP*, Seattle, Washington, USA, p. 322-332, 2013.

Nivre J., « Algorithms for Deterministic Incremental Dependency Parsing », *Computational Linguistics*, vol. 34, n° 4, p. 513-553, 2008.

Papineni K., Roukos S., Ward T., Zhu W.-J., « BLEU : A Method for Automatic Evaluation of Machine Translation », *Proc. ACL*, Philadelphia, Pennsylvania, p. 311-318, 2002.

Puterman M. L., *Markov Decision Processes : Discrete Stochastic Dynamic Programming*, 1st edn, John Wiley & Sons, Inc., 1994.

Ranzato M., Chopra S., Auli M., Zaremba W., « Sequence Level Training with Recurrent Neural Networks », *Proc. ICLR*, 2016.

Ross S., Bagnell J. A., « Reinforcement and Imitation Learning via Interactive No-Regret Learning », *arXiv preprint arXiv :1406.5979v1*, 2014.

Ross S., Gordon G., Bagnell D., « A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning », *Proc. AIStats*, vol. 15, Fort Lauderdale, FL, p. 627-635, 2011.

Shen L., Satta G., Joshi A., « Guided Learning for Bidirectional Sequence Classification », *Proc. ACL*, Prague, Czech Republic, p. 760-767, 2007.

Smith N. A., *Linguistic Structure Prediction*, 1st edn, Morgan & Claypool Publishers, 2011.

Sokolov A., Wisniewski G., Yvon F., « Lattice BLEU Oracles in Machine Translation », *ACM Transactions on Speech and Language Processing*, vol. 10, n° 4, p. 18 :1-18 :29, 2013.

Sun W., Venkatraman A., Gordon G. J., Boots B., Bagnell J. A., « Deeply AggreVaTeD : Differentiable Imitation Learning for Sequential Prediction », *in* D. Precup, Y. W. Teh (eds), *Proc. ICML*, vol. 70, Sydney, Australia, p. 3309-3318, 2017.

Sutton R. S., Barto A. G., *Introduction to Reinforcement Learning*, 1st edn, MIT Press, 1998.

Tellier I., Tommasi M., « Champs Markoviens Conditionnels pour l'extraction d'information », *in* E. Gaussier, F. Yvon (eds), *Modèles probabilistes pour l'accès à l'information textuelle*, Hermès, p. 223-267, 2011.

Venkatraman A., Hebert M., Bagnell J. A., « Improving Multi-step Prediction of Learned Time Series Models », *Proc. AAAI*, Austin, TX, p. 3024-3030, 2015.

Viterbi A. J., « Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm », *IEEE Trans. on Information Theory*, vol. 13, n° 2, p. 260-269, 1967.

Vlachos A., Craven M., « Search-based Structured Prediction Applied to Biomedical Event Extraction », *Proc. CoNLL*, Portland, OR, p. 49-57, 2011.

Wiseman S., Rush A. M., « Sequence-to-Sequence Learning as Beam-Search Optimization », *Proc. EMNLP*, Austin, TX, p. 1296-1306, November, 2016.

Wiseman S., Rush A. M., Shieber S. M., « Learning Global Features for Coreference Resolution », *Proc. NAACL-HLT*, San Diego, CA, p. 994-1004, 2016.

Zhang Y., Nivre J., « Transition-based Dependency Parsing with Rich Non-local Features », *Pro. ACL-HLT*, Portland, Oregon, p. 188-193, 2011.

Approche supervisée à base de cellules LSTM bidirectionnelles pour la désambiguïsation lexicale

Loïc Vial — Benjamin Lecouteux — Didier Schwab

Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, 38000 Grenoble, France
{loic.vial,benjamin.lecouteux,didier.schwab}@univ-grenoble-alpes.fr

RÉSUMÉ. *En désambiguïsation lexicale, l'utilisation des réseaux de neurones est encore peu présente et très récente. Cette direction est pourtant très prometteuse, tant les résultats obtenus par ces premiers systèmes arrivent systématiquement en tête des campagnes d'évaluation, malgré une marge d'amélioration qui semble encore importante. Nous présentons dans cet article une nouvelle architecture à base de réseaux de neurones pour la désambiguïsation lexicale. Notre système est à la fois moins complexe à entraîner que les systèmes neuronaux existants et il obtient des résultats état de l'art sur la plupart des tâches d'évaluation de la désambiguïsation lexicale en anglais. L'accent est porté sur la reproductibilité de notre système et de nos résultats, par l'utilisation d'un modèle de vecteurs de mots, de corpus d'apprentissage et d'évaluation librement accessibles.*

ABSTRACT. *In Word Sense Disambiguation, there are still few usages of neural networks. This direction is very promising however, the results obtained by these first systems being systematically in the top of the evaluation campaigns, with a scope for improvement which seems still large. We present in this paper a new architecture based on neural networks for Word Sense Disambiguation. Our system is at the same time less difficult to train than existing neural networks, and it obtains state of the art results on most evaluation tasks in English. The focus is on the reproducibility of our system and our results, through the use of a word embeddings model, training corpora and evaluation corpora freely accessible.*

MOTS-CLÉS : *désambiguïsation lexicale, approche supervisée, LSTM, réseau neuronal.*

KEYWORDS: *Word Sense Disambiguation, Supervised Approach, LSTM, Neural Network.*

1. Introduction

La désambiguïsation lexicale (DL) est une tâche centrale en traitement automatique des langues (TAL) qui vise à attribuer le sens le plus probable à un mot donné dans un document, à partir d'un inventaire prédéfini de sens.

Il existe une multitude d'approches pour la DL, dont les approches supervisées, qui utilisent des méthodes d'apprentissage automatique couplées à de grandes quantités de données manuellement annotées, les approches à base de connaissances, qui s'appuient sur des ressources lexicales telles que des dictionnaires, des thésaurus ou des réseaux lexicaux par exemple, les approches semi-supervisées, non supervisées, ou encore les approches à base de graphes ou de similarités. Pour un état de l'art plus complet, le lecteur est invité à lire par exemple Navigli (2009).

Depuis la création des campagnes d'évaluation pour les systèmes de DL telles que SensEval-SemEval, les approches supervisées se retrouvent systématiquement dans les premières places en termes de scores obtenus (Iacobacci *et al.*, 2016). Alors que les utilisations de techniques d'apprentissage à base de réseaux de neurones se multiplient dans la plupart des champs de recherche du TAL, par exemple pour la représentation vectorielle des mots ou la traduction automatique, on retrouve aussi des approches supervisées à base de réseaux de neurones pour la désambiguïsation lexicale, et ce sont ces méthodes qui obtiennent aujourd'hui les résultats état de l'art (Yuan *et al.*, 2016 ; Kågebäck et Salomonsson, 2016 ; Raganato *et al.*, 2017).

Dans cet article, nous présentons une nouvelle approche supervisée de DL à base de réseaux de neurones, qui s'appuie sur les modèles existants et qui obtient des résultats état de l'art sur la plupart des tâches d'évaluation de la DL en anglais tout en étant moins complexe et difficile à mettre en place. De plus, nous utilisons pour la première fois l'ensemble des corpus annotés avec des sens provenant de la base lexicale WordNet (Miller, 1995), ce qui permet à notre système d'être plus robuste car plus généralisable à de nouvelles données.

En effet, les systèmes supervisés sont généralement uniquement entraînés sur le SemCor (Miller *et al.*, 1993), mais une demi-douzaine d'autres corpus annotés en sens et de grande taille existent, et ils ont été récemment regroupés dans une ressource libre nommée UFSAC[1] (Vial *et al.*, 2018). Par souci de comparaison avec les systèmes état de l'art, nous avons évalué notre approche à la fois en utilisant tous les corpus UFSAC disponibles, mais aussi en nous restreignant uniquement au SemCor.

Dans un premier temps, nous allons présenter une vue d'ensemble de la désambiguïsation lexicale ainsi que les architectures des systèmes neuronaux de DL de l'état de l'art, avec leurs avantages et inconvénients respectifs dans la section 2, avec une description et une comparaison des corpus d'entraînement pour les systèmes supervisés. Puis nous présenterons notre architecture dans la section 3. Nous décrirons ensuite le protocole expérimental suivi pour évaluer notre système dans la section 4 puis nous

1. `https://github.com/getalp/UFSAC`

détaillerons les résultats dans la section 5 avec une analyse des erreurs. Enfin nous présenterons un travail préliminaire d'amélioration de notre système de manière totalement non supervisée dans la section 6 et enfin nous conclurons dans la section 7.

2. Désambiguïsation lexicale

La désambiguïsation lexicale est une tâche visant à assigner le sens le plus probable des mots contenus dans un texte, à partir d'un inventaire prédéfini de sens. Par exemple, si l'on considère la phrase « La souris mange le fromage. », un bon système de désambiguïsation lexicale devrait plutôt assigner au mot « souris » le sens du rongeur, plutôt que le sens du périphérique informatique.

Cette façon de clarifier explicitement un texte en y assignant un label de sens à chacun de ces mots a montré son efficacité dans l'amélioration de diverses tâches. Dans les travaux de Zhong et Ng (2012) par exemple, un système de DL est intégré à un système de traduction automatique statistique et permet d'améliorer ses performances. Plus récemment, les travaux de Rios *et al.* (2017) et Liu *et al.* (2018) ont permis de montrer les difficultés pour un système de traduction automatique neuronale de faire un choix lorsqu'il est confronté à un mot polysémique, et ont intégré avec succès les prédictions d'un système de DL pour améliorer ses performances. Enfin, les travaux de Chan *et al.* (2007a) ont montré que l'utilisation des prédictions d'un système de DL pour la création d'un modèle de langue pour la recherche d'informations améliore de façon significative un système état de l'art sur cette tâche.

2.1. *Approches pour la désambiguïsation lexicale*

Les approches pour la désambiguïsation lexicale sont multiples et sont généralement classées en plusieurs catégories en fonction de la nature des ressources utilisées et de leur quantité. On distingue ainsi :

– les approches à base de connaissances, qui s'appuient principalement sur des ressources telles que des dictionnaires, réseaux lexicaux ou graphes par exemple. On retrouve ici les approches à base de mesures de similarité sémantiques comme celles exploitant l'algorithme de Lesk (Lesk, 1986), ou les mesures exploitant un graphe comme celui de BabelNet (Navigli et Ponzetto, 2010) ;

– les approches supervisées, qui exploitent un grand nombre d'exemples de mots annotés manuellement ou automatiquement en sens, et qui sont généralement liés à une méthode d'apprentissage automatique telle qu'un classifieur linéaire (Chan *et al.*, 2007b ; Zhong et Ng, 2010) ou plus récemment un réseau de neurones récurrents (Yuan *et al.*, 2016 ; Raganato *et al.*, 2017) ;

– les approches non supervisées, qui réalisent de l'induction de sens, c'est-à-dire que seuls des textes non annotés sont utilisés, et les différents sens des mots sont induits en fonction de leurs contextes (Brody et Lapata, 2008 ; Yarowsky, 1995).

Bien sûr les frontières entre ces catégories sont parfois floues, on retrouve par exemple des approches à base de connaissances améliorées à l'aide de corpus annotés

en sens (Vial *et al.*, 2016), une catégorie de méthodes dites faiblement supervisées qui visent l'utilisation d'un minimum de données annotées en sens, et des approches semi-supervisées qui exploitent des données annotées ainsi que des données non annotées dans leurs approches (Yuan *et al.*, 2016).

2.2. *Évaluation des systèmes de désambiguïsation lexicale*

L'évaluation des différents systèmes de DL est rendue possible notamment grâce à la création de la première campagne d'évaluation nommée SensEval (Kilgarriff, 1998), centrée sur la tâche de DL. Après deux autres éditions de SensEval, cette campagne a été renommée en SemEval et touche maintenant à des tâches diverses allant de la reconnaissance d'entités nommées à de l'analyse de sentiments. Les tâches de DL sont pour autant toujours régulièrement présentes (Edmonds et Cotton, 2001 ; Snyder et Palmer, 2004 ; Navigli *et al.*, 2007 ; Navigli *et al.*, 2013 ; Moro et Navigli, 2015).

Dans ces tâches de DL, les systèmes comparés doivent annoter en sens tous les mots d'un document ou une partie d'entre eux. Les annotations sont ensuite comparées aux références et les performances des systèmes sont mesurées selon les métriques couverture (C), précision (P), rappel (R) et F-mesure (F1).

La principale langue cible de ces tâches d'évaluation est l'anglais, mais il existe aussi des tâches pour la désambiguïsation d'autres langues comme le français ou l'italien, par exemple (Navigli *et al.*, 2013 ; Moro et Navigli, 2015). Toutefois, une des raisons principales pour expliquer la prédominance de l'anglais comme langue cible de la désambiguïsation lexicale est l'existence de WordNet (Miller, 1995), une base de données lexicale pour l'anglais de grande qualité, utilisée souvent comme l'inventaire de sens de référence pour annoter les documents.

2.3. *Architectures neuronales pour la désambiguïsation lexicale*

On retrouve dans les approches neuronales pour la DL notamment trois travaux majeurs : le modèle de Kågebäck et Salomonsson (2016), le modèle de Yuan *et al.* (2016) et celui de Raganato *et al.* (2017).

Kågebäck et Salomonsson (2016) sont les premiers à mettre en œuvre un réseau de neurones à base de vecteurs de mots et de cellules récurrentes de type LSTM pour prédire le sens d'un mot cible. Dans leurs travaux, un modèle n'est capable de prédire le sens que d'un seul lemme du dictionnaire, et donc chaque lemme a son modèle propre de classification qui est entraîné séparément. Leur système est évalué sur les tâches de *lexical sample* des campagnes d'évaluation SensEval 2 et SensEval 3 dans lesquelles plusieurs instances d'un faible nombre de lemmes distincts sont à annoter en sens, mais il n'est pas évalué sur les tâches de désambiguïsation lexicale *all words* où tous les mots d'un document doivent être annotés en sens.

Le principal avantage de leur modèle est donc sa petite taille. En effet la couche de sortie de leur réseau est de la taille du nombre de sens pour le lemme cible, le

nombre de sens moyen pour les mots polysémiques dans WordNet étant d'environ 3[2]. Les couches cachées de cellules LSTM sont, elles aussi, très petites, avec seulement deux couches de taille 74 chacune. Il est cependant peu aisé d'entraîner ce système à annoter tous les mots d'un document car chaque lemme doit avoir son propre modèle.

Dans le modèle de Yuan *et al.* (2016), un réseau neuronal à base de cellules LSTM est utilisé comme modèle de langue, pour prédire un mot d'une séquence en fonction de son contexte. Un apprentissage supervisé sur des corpus annotés en sens est ensuite effectué pour que leur système apprenne à distinguer les différents sens d'un mot en fonction des mots prédits par leur modèle de langue. Dans un second temps, les auteurs proposent une méthode de propagation de labels pour augmenter leurs données annotées en sens et obtenir ainsi leurs meilleurs résultats. Cette méthode consiste à chercher dans des corpus non annotés de nouvelles phrases, proches des phrases de leur corpus annoté, en s'appuyant sur sur une mesure de similarité cosinus entre les représentations vectorielles de ces phrases. Les annotations en sens sont ensuite propagées de la phrase initialement annotée vers l'autre phrase.

Dans cet article, les auteurs comparent les performances de différents modèles, entraînés sur le SemCor ou l'OMSTI, avec et sans leur propagation de labels, et obtiennent des résultats état de l'art sur la plupart des tâches. Le principal problème de leur approche est la reproductibilité des résultats. En effet, leur modèle de langue est entraîné sur un corpus privé d'actualités (*news*) d'une taille de 100 milliards de mots, et ils ont utilisé pour leur propagation de labels des phrases prises aléatoirement sur le Web, sans en spécifier la source plus précisément.

Enfin, l'architecture de leur modèle de langue ne permet de prédire le sens que d'un seul mot à la fois pour une séquence donnée. Il est donc nécessaire d'exécuter leur modèle pour chaque mot d'une phrase afin de tous les annoter.

Raganato *et al.* (2017) proposent également un modèle à base de LSTM mais qui prédit directement un label pour chacun des mots donnés en entrée. Ce label fait partie d'un ensemble comprenant tous les sens d'un dictionnaire ainsi que tous les mots observés pendant l'entraînement. Ils augmentent ensuite leur modèle avec une couche d'attention et effectuent un entraînement multitâche dans lequel leur réseau prédit à la fois un sens ou un mot, un label de partie du discours et un label sémantique.

Cette architecture est la seule permettant d'annoter tous les mots d'une séquence en une passe et l'entraînement de leur modèle s'est effectué sur le SemCor uniquement. Leur réseau associe à un mot en entrée un label appartenant à l'ensemble des sens de leur inventaire de sens ainsi que l'ensemble des mots observés pendant l'entraînement. Cette approche permet à leur modèle d'apprendre à prédire un label de sens lorsque le mot est annoté dans le corpus d'entraînement, et un label de mot lorsque le mot n'est pas annoté (si c'est un mot-outil par exemple). L'inconvénient de l'approche est qu'elle n'est pas applicable pour l'apprentissage sur un corpus partiellement annoté en

2. `https://wordnet.princeton.edu/documentation/wnstats7wn`

sens. En effet, pour ce type de corpus, leur modèle va apprendre à recopier des mots non annotés, au lieu d'essayer de les annoter.

2.4. *Corpus annotés en sens pour l'entraînement de systèmes supervisés*

Les corpus annotés en sens sont la matière première des approches supervisées, et leur utilisation ou non en tant que données d'entraînement est ainsi un choix crucial.

Le SemCor (Miller *et al.*, 1993) est indéniablement le corpus le plus utilisé en tant que données d'entraînement dans les systèmes supervisés. Il est en effet la source principale de données annotées des meilleurs systèmes depuis la création des premières campagnes d'évaluation (Hoste *et al.*, 2001 ; Decadt *et al.*, 2004) jusqu'aux systèmes les plus récents (Yuan *et al.*, 2016 ; Raganato *et al.*, 2017).

D'autres corpus sont aussi utilisés dans un plus faible nombre de travaux. C'est le cas, par exemple, du DSO (Ng et Lee, 1997) qui est utilisé en plus du SemCor dans le système NUS-PT (Chan *et al.*, 2007b), le meilleur système de la campagne d'évaluation SemEval 2007 (Navigli *et al.*, 2007) ou de l'OMSTI (Taghipour et Ng, 2015), un corpus d'un million de mots annotés en sens, utilisé notamment par Iacobacci *et al.* (2016) en remplacement du SemCor et qui permet aux auteurs d'obtenir leurs meilleurs résultats. On peut aussi citer le MASC (Ide *et al.*, 2008), un corpus utilisé dans le récent travail de Yuan *et al.* (2016) pour l'évaluation de leur système, l'OntoNotes (Hovy *et al.*, 2006) ou encore le WordNet Gloss Tagged (Miller, 1995).

On remarque ainsi qu'aucun travail n'utilise l'ensemble des corpus annotés en sens disponibles, et rares sont ceux qui justifient l'emploi d'un corpus plutôt qu'un autre. Jusqu'à récemment, ces ressources pouvaient être difficiles d'accès et d'utilisation, notamment à cause des différents inventaires de sens utilisés (différentes versions de WordNet, ou même d'autres bases lexicales), ou bien des différents formats de fichiers. Cependant, les travaux de Vial *et al.* (2018) apportent une ressource nommée UFSAC contenant l'ensemble des corpus annotés en sens cités précédemment, dans un format et un inventaire de sens unifié, ce qui facilite leur utilisation.

Dans cet article, nous proposons pour la première fois d'entraîner un système supervisé à partir de ces données. Nous présentons aussi une synthèse des différences notables entre ces corpus dans le tableau 1. Ces chiffres font ainsi ressortir les caractéristiques propres de chaque corpus.

L'OMSTI est le plus grand corpus concernant le nombre de mots annotés et non annotés, mais il est celui qui a le plus faible ratio entre ces deux nombres : moins d'un mot sur trente est effectivement annoté en sens, là où pour le SemCor, près d'un mot sur quatre est annoté. L'OntoNotes, le DSO et le MASC ont aussi cette caractéristique d'avoir un faible ratio de mots annotés, alors que le WordNet Gloss Tagged a, comme pour le SemCor, un ratio de plus d'un mot sur cinq annotés.

Pour ce qui est de la diversité des lemmes, le SemCor et le WNGT se démarquent là encore des quatre autres corpus. En effet, plus de 10 000 lemmes différents sont annotés dans le SemCor et près de 20 000 dans le WNGT, alors qu'à l'extrême opposé,

Corpus	SemCor	DSO	WNGT	MASC	OMSTI	OntoNotes	Tous
Nombre de mots, en milliers	779 (#5) (1,77 %)	2 705 (#2) (6,15 %)	1 635 (#4) (3,72 %)	585 (#6) (1,33 %)	35 800 (#1) (81,40 %)	2 476 (#3) (5,63 %)	43 980
Nombre de mots annotés en sens, en milliers	189 (#3) (10,94 %)	176 (#4) (10,17 %)	329 (#2) (19,02 %)	45 (#6) (2,58 %)	917 (#1) (52,93 %)	75 (#5) (4,35 %)	1 732
Ratio entre le nombre de mots annotés et le nombre de mots total	0,24 (#1)	0,07 (#4)	0,20 (#2)	0,08 (#3)	0,03 (#6)	0,03 (#5)	0,04
Nombre de lemmes distincts annotés en sens	11 646 (#2) (57,27 %)	191 (#6) (0,94 %)	19 150 (#1) (94,18 %)	3 767 (#3) (18,53 %)	1 140 (#5) (5,61 %)	2 124 (#4) (10,45 %)	20 334
Nombre d'exemples par lemme	16 (#5)	922 (#1)	17 (#4)	11 (#6)	804 (#2)	35 (#3)	85
Nombre d'exemples par sens	6,04 (#6)	175,46 (#2)	6,06 (#5)	9,24 (#4)	262,86 (#1)	20,10 (#3)	19,38
Nombre de sens différents par lemme *dans WordNet*	3,66 (#5)	10,62 (#1)	3,18 (#6)	4,41 (#4)	6,93 (#2)	5,35 (#3)	3,14
Nombre de sens différents par lemme *dans le corpus*	1,94 (#4)	7,35 (#1)	2,27 (#3)	1,25 (#6)	3,26 (#2)	1,69 (#5)	2,40
Ratio entre le nombre de sens différents par lemme dans le corpus et le nombre de sens différents par lemme dans WordNet	0,56 (#3)	0,74 (#1)	0,72 (#2)	0,36 (#6)	0,55 (#4)	0,42 (#5)	0,75

Tableau 1. *Statistiques des corpus d'entraînement de la ressource UFSAC. Les chiffres entre parenthèses préfixés par un # sont les rangs, ceux suivis par un % sont la part de ce corpus par rapport au corpus « Tous ». Le corpus « Tous » correspond à un corpus constitué de la concaténation de tous les autres.*

le DSO ne contient que 191 lemmes différents. À noter que le nombre de lemmes polysémiques dans WordNet 3.0 s'élève à 26 896[3].

Dans les cinquième, sixième et septième lignes du tableau, on fait émerger une des caractéristiques principales de ces corpus : le DSO et l'OMSTI sont en effet construits autour d'un nombre restreint de lemmes très fortement polysémiques, mais avec un très grand nombre d'exemples pour chaque lemme et chaque sens de ces lemmes. À l'opposé, le MASC, l'OntoNotes et surtout le SemCor et le WNGT sont construits plutôt autour d'un certain type de documents (actualités, définitions d'un dictionnaire...) et donc les lemmes sont plus variés, avec un nombre de sens plus variable et un nombre d'exemples par lemme plus réduit.

Enfin les deux dernières lignes font ressortir une dernière caractéristique de ces corpus : à quel point tous les sens de chaque lemme sont représentés au sein du corpus, par rapport à ceux présents dans WordNet. En effet, un ratio inférieur à 1 dans la dernière ligne indique que pour chacun des lemmes présents dans le corpus, tous ses différents sens existants dans le dictionnaire ne sont pas représentés dans le corpus. On peut même dire que pour le MASC et l'OntoNotes, par exemple, en moyenne moins de la moitié des sens possibles d'un lemme sont représentés dans le corpus, ce qui pourrait poser problème à un système supervisé pour généraliser à de nouvelles données ce qu'il a appris sur ces corpus. Le DSO et le WNGT ont eux en moyenne près de trois quarts des sens représentés par lemmes présents dans leurs données.

Outre le fait de mieux comprendre quelles sont les différences marquées entre les corpus d'entraînement de la ressource UFSAC, ces chiffres nous permettent de voir l'intérêt de tous les combiner pour l'apprentissage d'un système supervisé de désambiguïsation lexicale. En effet, la colonne « Tous » montre les mêmes statistiques vues précédemment, appliquées à cet ensemble constitué de tous les corpus et on voit bien, par exemple sur la quatrième ligne ou sur la dernière ligne, que cette combinaison constitue un ensemble riche de données annotées, avec plus de 20 000 lemmes uniques présents, et le plus grand ratio de polysémie moyenne dans le corpus des lemmes représentés, sur la polysémie moyenne de ces lemmes dans WordNet. Cependant, cette combinaison donne aussi un des ratios de nombre de mots annotés sur le nombre de mots total parmi les plus bas. Les phrases quasi entièrement annotées du SemCor et du WNGT se retrouvent donc un peu noyées dans les phrases faiblement annotées de l'OMSTI ou de l'OntoNotes.

Afin de voir l'impact du choix du corpus d'apprentissage sur les performances du système que nous proposons, nous allons utiliser comme données d'entraînement cet ensemble de tous les corpus UFSAC d'une part, mais nous allons aussi nous restreindre aux données du SemCor uniquement, afin de nous comparer aux systèmes de l'état de l'art entraînés uniquement sur ce corpus.

3. `https://wordnet.princeton.edu/documentation/wnstats7wn`

3. Nouvelle architecture neuronale pour la désambiguïsation lexicale

Notre approche est, comme pour Raganato *et al.* (2017), de considérer la désambiguïsation lexicale comme un problème de classification dans lequel un label est assigné à chaque mot. Nous simplifions cependant leur modèle en considérant qu'un label appartient uniquement à l'ensemble de tous les sens de notre inventaire de sens. L'architecture de notre réseau de neurones repose ainsi sur trois couches de cellules.

La première couche, ou couche d'entrée, permet d'associer à un mot une forme vectorielle à l'aide d'une table de correspondance construite séparément de notre système. On pourra utiliser ici n'importe quelle base de vecteurs de mots pré-entraînés telle que Word2Vec (Mikolov *et al.*, 2013) ou GloVe (Pennington *et al.*, 2014). Si un mot est absent de la base de vecteurs utilisée, alors on lui associe le vecteur nul.

Ensuite, la couche cachée, composée de cellules LSTM (Hochreiter et Schmidhuber, 1997) bidirectionnelles, va calculer une nouvelle représentation vectorielle pour chacun des mots en fonction des mots précédents et suivants. En effet, ces cellules dites « à mémoire » aussi appelées cellules « récurrentes » permettent de calculer une sortie en considérant non seulement l'élément courant de la séquence, mais aussi l'historique passé des cellules précédentes. Nous utilisons l'implémentation d'une cellule LSTM incluse dans l'outil PyTorch[4] formulée ainsi :

$$
\begin{array}{ll}
i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi}) & f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf}) \\
g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg}) & o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho}) \\
c_t = f_t c_{(t-1)} + i_t g_t & h_t = o_t \tanh(c_t)
\end{array}
$$

La sortie h_t d'une cellule LSTM à l'instant t est donc fonction de son entrée x_t, de ses états précédents $h_{(t-1)}$ et $c_{(t-1)}$, des matrices de poids W_i et W_h et des vecteurs de biais b_i et b_h. Ces poids et ces biais étant les paramètres qui vont être appris par notre modèle pendant l'entraînement. Enfin, σ est la fonction sigmoïde.

Pour obtenir une sortie de cellule LSTM bidirectionnelle, nous appliquons ces formules pour chacune des deux directions (t allant de 0 à n pour une direction, et de n à 0 pour l'autre direction), et nous concaténons leur sortie.

Enfin, la dernière couche de notre architecture, la couche de sortie, applique une transformation linéaire et une fonction softmax classique à la sortie du LSTM, afin de générer pour chacun des mots en entrée, une distribution de probabilité sur tous les sens possibles du dictionnaire. Nous suivons ainsi la formule $y = softmax(Ax + b)$, avec x la sortie du LSTM, A la matrice de poids et b le vecteur de biais, A et b étant des paramètres de notre modèle.

La fonction de coût à minimiser pendant la phase d'apprentissage est l'entropie croisée entre la couche de sortie et un vecteur de type *one-hot*, pour lequel toutes les composantes sont à 0 sauf à l'index du sens cible où elle est à 1. On cherche ainsi à minimiser la fonction $H(p, q) = -\sum_x p(x) \log q(x)$, où x est une composante du vecteur de la couche de sortie, p est la distribution de probabilité réelle et q la sortie

4. `https://pytorch.org`

de notre réseau de neurones. Comme toutes les valeurs de la distribution réelle sont à 0 sauf à l'index du sens correct, pour un exemple donné, on cherche ainsi à minimiser la formule $-\log q(s)$, où s est l'index du sens à prédire.

Notre modèle prédit toujours un sens en sortie pour chaque mot en entrée, même pour les mots-outils ou les mots qui n'ont pas été annotés dans le corpus d'entraînement. Cependant, dans ces cas-là, nous annotons le mot avec un symbole spécial `<skip>` qui sert de marqueur pendant l'entraînement en indiquant d'ignorer les prédictions faites par le modèle et de ne pas en tenir compte lors de la rétropropagation. Plus précisément, la fonction de coût n'est pas calculée quand ce marqueur est présent et donc le gradient n'est pas mis à jour.

Contrairement à l'approche proposée par Raganato *et al.* (2017), nous entraînons notre modèle non seulement sur des données entièrement annotées, comme c'est le cas avec le SemCor par exemple, mais également sur des données partiellement annotées, comme l'OMSTI, dans lequel un seul mot est annoté par phrase. Il est en effet capable d'apprendre à prédire les sens de tous les mots annotés dans une séquence en même temps, tout en ignorant les éléments non annotés.

De plus, notre modèle a l'avantage par rapport à celui de Raganato *et al.* (2017) d'avoir un nombre de paramètres réduit à la taille du vocabulaire de sortie. En effet, notre modèle prédit un label appartenant à l'ensemble des sens vus pendant l'entraînement, ce qui correspond à un ensemble de taille 26 215 en entraînant sur le SemCor, et 70 131 en utilisant tous les corpus UFSAC. L'approche de Raganato *et al.* (2017), quant à elle, inclut dans le vocabulaire de sortie du modèle, en plus de tous les sens, tous les *mots* vus pendant l'entraînement et qui sont présents dans le modèle de vecteurs de mots utilisé en entrée. Les auteurs ne fournissent pas de chiffre ni leur code, mais nous estimons en utilisant le même modèle de vecteurs de mots qu'eux, et en filtrant sur les mots présents dans le SemCor (leur corpus d'entraînement), que ce sont 34 980 labels qui sont ajoutés à leur vocabulaire de sortie. En appliquant cette méthode sur tous les corpus d'entraînement de UFSAC, 144 308 labels supplémentaires seraient ajoutés.

Notre architecture est enfin très différente de celles de Yuan *et al.* (2016) ou de Kågebäck et Salomonsson (2016), notamment car elles ne permettent pas d'annoter tous les mots en entrée de leurs modèles en une seule passe.

4. Protocole expérimental

Dans cette section, nous détaillons le processus que nous avons suivi concernant l'apprentissage et l'évaluation de notre système de DL à base de réseau neuronal.

4.1. *Corpus d'entraînement, de développement et d'évaluation*

Nous avons tiré parti du travail de Vial *et al.* (2018) et décidé d'utiliser pour la première fois les six corpus d'entraînement de la ressource UFSAC décrits dans la sous-section 2.4, à savoir : le SemCor, le DSO, le WordNet Gloss Tagged, l'OMSTI,

le MASC et l'OntoNotes. De plus, afin de comparer l'architecture que nous proposons avec l'état de l'art, et notamment Raganato *et al.* (2017) et Yuan *et al.* (2016), qui utilisent uniquement le SemCor comme corpus d'apprentissage supervisé, nous avons aussi évalué notre approche en limitant l'apprentissage du modèle à ce corpus.

Nous avons utilisé le corpus de la tâche 13 de SemEval 2015 (Moro et Navigli, 2015) comme corpus de développement durant l'apprentissage. Le modèle ayant obtenu le meilleur score F1 (sur le développement) a été utilisé sur les corpus de SensEval 2 (Edmonds et Cotton, 2001), SensEval 3 (Snyder et Palmer, 2004), les tâches 7 et 17 de SemEval 2007 (Navigli *et al.*, 2007 ; Pradhan *et al.*, 2007), et enfin la tâche 12 de SemEval 2013 (Navigli *et al.*, 2013).

Dans certains corpus, les mots peuvent être annotés avec plusieurs sens WordNet, soit parce que l'annotateur a trouvé qu'ils étaient tous applicables, ou bien parce qu'ils ont été initialement annotés avec un autre dictionnaire puis les sens ont été convertis en sens WordNet. Dans ce cas nous supprimons entièrement l'annotation du mot pour ne garder au final que ceux qui ont été annotés avec un seul sens dans notre corpus d'apprentissage. Pour le SemCor, le DSO, le WordNet Gloss Tagged et l'OMSTI, ce cas ne concerne respectivement que 0,38 %, 0,02 %, 0 % et 0,02 % des annotations. Cependant pour le MASC et l'OntoNotes, qui ont été initialement annotés avec un dictionnaire distinguant moins finement les sens que WordNet, ce sont respectivement 55,98 % et 65,93 % des annotations qui sont enlevées.

Afin de mesurer l'impact du mélange des corpus d'entraînement ayant des proportions de mots annotés très différentes, nous avons mené l'expérience qui consiste à répartir les annotations du SemCor en plusieurs phrases, et de mesurer les répercutions sur notre système. Nous avons créé deux nouvelles versions du SemCor. Une version « éclatée » à 100 % dans laquelle pour chaque phrase contenant n annotations, nous dupliquons cette phrase n fois et annotons un seul mot différent dans chacune d'elles. Puis une version « éclatée » à 50 % dans laquelle nous appliquons cette opération sur seulement la moitié des phrases du SemCor, puis mélangeons toutes les phrases. Nous avons ainsi entraîné huit modèles sur chacun de ces corpus, et les résultats de nos systèmes sur le corpus de développement se trouvent dans le tableau suivant :

Données d'entraînement	Nombre de phrases	Score F1 (%)	Écart-type
SemCor original	36 321	72,24	$\pm$ 0,64
SemCor « éclaté » à 50 %	132 945	71,50	$\pm$ 0,62
SemCor « éclaté » à 100 %	227 993	71,72	$\pm$ 0,61

Bien qu'une tendance se dégage en faveur de la version originale du SemCor, même en se plaçant dans le cas extrême d'un seul mot annoté par phrase, l'impact sur le score final est négligeable. Notre architecture est donc capable d'apprendre sur des phrases partiellement annotées, sans dégradation notable.

4.2. *Hyperparamètres du modèle neuronal*

En entrée de notre réseau, nous avons utilisé les vecteurs de GloVe (Pennington *et al.*, 2014) pré-entraînés sur Wikipedia 2014 et Gigaword 5 disponibles librement[5]. La taille des vecteurs est de 300, la taille du vocabulaire est de 400 000 et tous les mots sont en minuscules. Nous avons choisi ces vecteurs pour la petite taille de leur modèle pré-entraîné et pour leur qualité démontrée par les auteurs sur les tâches de similarité de mots et d'analogie de mots. Ce sont aussi ces vecteurs qui sont utilisés en entrée du réseau décrit par Kågebäck et Salomonsson (2016).

Pour la couche cachée de neurones récurrents, nous avons choisi des cellules LSTM de taille 1 000 par direction (donc 2 000 au total). C'est, à peu de choses près, la taille qui est utilisée dans les travaux de Yuan *et al.* (2016) (une couche de taille 2 048), et c'est environ deux fois moins que ce qui est utilisé par Raganato *et al.* (2017) (deux couches de cellules LSTM bidirectionnelles de taille 1 024).

Enfin, entre la couche cachée et la couche de sortie, nous avons appliqué une régularisation *Dropout* (Srivastava *et al.*, 2014) à $50\,\%$. Cette méthode vise à empêcher le surapprentissage pendant l'entraînement afin de rendre le modèle plus robuste.

Système	Vecteurs en entrée	Couche LSTM	Taux de *Dropout*	Score F1 (%)	Écart-type
(Raganato *et al.*, 2017)	Word2Vec	2*1 024	0	69,2	-
Notre système (SemCor, moyenne de huit modèles entraînés séparément)	Word2Vec	2*1 024	0	68,76	$\pm 0{,}77$
	Word2Vec	2*1 024	0,5	68,47	$\pm 0{,}57$
	Word2Vec	1*1 000	0	69,06	$\pm 0{,}71$
	Word2Vec	1*1 000	0,5	69,25	$\pm 0{,}48$
	GloVe	2*1 024	0	71,21	$\pm 0{,}49$
	GloVe	2*1 024	0,5	72,08	$\pm 0{,}60$
	GloVe	1*1 000	0	71,37	$\pm 0{,}35$
	GloVe	1*1 000	0,5	72,24	$\pm 0{,}64$

Tableau 2. *Résultats obtenus par notre système entraîné sur le SemCor et celui de Raganato et al. (2017), sur notre corpus de développement (SemEval 2015 tâche 13) en fonction des hyperparamètres du modèle neuronal.*

Dans le tableau 2, nous montrons l'impact de ces hyperparamètres sur l'apprentissage de notre modèle en faisant varier chacun d'eux entre les valeurs que nous avons choisies et celles utilisées par Raganato *et al.* (2017).

Comme nous pouvons le voir, à paramètres équivalents, c'est-à-dire en utilisant le même modèle de vecteurs de mots, avec deux couches de LSTM et sans *Dropout*, notre système obtient des résultats qui ne sont pas significativement différents de ceux de Raganato *et al.* (2017). De même, réduire de moitié le nombre de couches de cellules LSTM ne fait pas varier significativement les résultats en diminuant pourtant largement le nombre de paramètres du modèle. La différence de résultats est cependant

5. `https://nlp.stanford.edu/projects/glove/`

notable lorsque l'on utilise les vecteurs GloVe plutôt que Word2Vec, avec systématiquement entre 2 % et 4 % d'amélioration. Enfin, l'utilisation du *Dropout* améliore les résultats de presque 1 % lorsque l'on utilise GloVe.

4.3. *Entraînement du modèle neuronal*

Les paramètres utilisés pour l'apprentissage sont les suivants : la méthode d'optimisation est Adam (Kingma et Ba, 2015), avec les mêmes paramètres par défaut tels que décrits dans leur article, la taille de mini-lots utilisée est de 30, les phrases sont tronquées à 50 mots, pour faciliter l'entraînement tout en minimisant la perte d'informations (moins de 5 % des mots annotés dans nos données d'entraînement sont perdus), enfin, les séquences sont remplies de vecteurs nuls depuis la fin de façon à ce qu'elles aient toutes la même taille au sein d'un mini-lot.

Nous avons construit notre réseau neuronal à l'aide de l'outil PyTorch[6] et nous avons effectué l'apprentissage pendant 20 cycles. Un cycle correspondant à une passe complète sur nos données d'entraînement. Nous avons évalué périodiquement (tous les 2 000 mini-lots et à la fin de chaque cycle) notre modèle sur le corpus de développement, et conservé uniquement celui ayant obtenu le plus grand score F1. Le code source de notre système ainsi que nos modèles pré-entraînés sont disponibles à l'adresse suivante : `https://github.com/getalp/disambiguate`.

4.4. *Processus de désambiguïsation*

Pour réaliser la désambiguïsation d'une séquence de mots en utilisant le réseau entraîné, chaque mot est d'abord transformé en vecteur à l'aide du modèle de vecteurs de mots, puis donné en entrée au réseau. En sortie, une distribution de probabilité sur tous les sens observés pendant l'apprentissage est retournée pour chaque élément de la séquence. Nous assignons le sens le plus probable en suivant cette distribution, parmi les sens possibles du mot dans WordNet, en fonction de son lemme et de sa partie du discours. Ces deux informations étant systématiquement données pendant les campagnes d'évaluation de la DL. Si aucun sens n'est assigné, une stratégie de repli est effectuée. La plus courante, et celle que nous utilisons, est d'assigner au mot son sens le plus fréquent dans WordNet.

Le processus d'apprentissage est forcément stochastique. En effet non seulement les poids du modèle sont initialisés aléatoirement par la bibliothèque sous-jacente, mais le corpus d'apprentissage est également mélangé à chaque début de cycle. Nous avons entraîné ainsi huit modèles séparément pour chacune de nos configurations, puis pour chacune de nos évaluations, nous donnons deux résultats :

– la moyenne des scores obtenus par chacun des huit modèles, ainsi que leur écart-type. Nous pouvons ainsi nous comparer avec les autres travaux de l'état de l'art qui n'entraînent qu'un seul modèle tout en ayant un moyen d'estimer la variabilité et la significativité des résultats ;

6. `http://pytorch.org/`

– le score d'un système « ensemble », c'est-à-dire un système qui moyenne les prédictions faites par ces huit modèles afin d'obtenir une nouvelle distribution de sens plus stable et généralement de meilleure qualité.

Pour le système « ensemble », nous avons utilisé une moyenne géométrique sur les prédictions faites par les modèles. Cette pratique couramment utilisée (par exemple Sutskever *et al.* (2014)) permet non seulement d'avoir un système moins sensible au bruit et donc plus robuste, mais aussi un système de meilleure qualité. En effet, un modèle peut être individuellement bloqué dans un minimum local pendant l'entraînement et avoir un bon score sur le corpus de développement, mais être incapable de généraliser, alors qu'il est improbable que ce problème arrive à l'ensemble de modèles.

5. Résultats

Nous avons évalué notre modèle sur tous les corpus d'évaluation des tâches de DL des campagnes d'évaluation SensEval-SemEval. Les scores obtenus par notre système comparés à ceux des systèmes semblables de l'état de l'art à base de réseaux de neurones (Yuan *et al.*, 2016 ; Raganato *et al.*, 2017), ainsi que l'étalon du sens le plus fréquent se trouvent dans le tableau 3.

On remarque d'abord qu'en termes de scores F1 avec le repli, il y a très peu de différences entre les modèles entraînés sur le SemCor et ceux entraînés sur tous les corpus UFSAC. Nos meilleurs résultats sur les tâches de SensEval 3 et de SemEval 2007 sont même obtenus par le système qui est entraîné sur le SemCor uniquement. Le SemCor possède pourtant seulement environ 10 % des mots annotés dans UFSAC.

Cependant, lorsque l'on compare les scores de désambiguïsation d'un système avec et sans repli, la différence entre ces deux scores est bien plus grande avec le système entraîné sur le SemCor qu'avec celui entraîné sur UFSAC. Ceci s'explique par la couverture du SemCor qui est moins importante que celle des corpus UFSAC réunis. Pour le système appris sur le SemCor, la couverture est en effet de 91 % sur SensEval 2, 97 % sur SensEval 3, 93 % sur SemEval 2007 (07), 98 % sur SemEval 2007 (17), 91 % sur SemEval 2013 et 95 % sur SemEval 2015. Pour celui appris sur tout UFSAC, la couverture est respectivement de 98 %, 99 %, 99 %, 99 %, 98 % et 99 %.

Ces résultats démontrent la grande qualité du SemCor, c'est en effet lorsque sa couverture sur les tâches d'évaluation est la plus proche de 100 % que notre système appris sur ce seul corpus obtient les meilleurs résultats. Les autres corpus UFSAC permettent quand même d'annoter un bien plus grand nombre de sens sans stratégie de repli, et nos meilleurs résultats sur SensEval 2, SemEval 2013 et SemEval 2015 sont obtenus avec le système appris sur tous les corpus réunis.

Le premier système de Yuan *et al.* (2016) obtient des résultats comparables aux nôtres mais comme nous l'avons souligné dans la section 2, le caractère privé de leur corpus d'entraînement contenant 100 milliards de mots pour leur modèle de langue rend très difficile la reproductibilité de leurs résultats.

Système	SE2	SE3	SE07 (07)	SE07 (17)	SE13 (12)	SE15 (13)
Sens le plus fréquent	65,6	66,0	78,89	54,5	63,8	67,1
Yuan *et al.* (2016) (LSTM)	73,6	69,2	82,8	**64,2**	67,0	72,1
Yuan *et al.* (2016) (LSTM + LP)	**73,8**	**71,8**	**83,6**	63,5	**69,5**	**72,6**
Raganato *et al.* (2017) (BLSTM)	71,4	68,8	-	*61,8	65,6	69,2
Raganato *et al.* (2017) (BLSTM + att. + LEX + POS)	72,0	69,1	83,1	*64,8	66,9	71,5
Notre système (SemCor, individuel, sans repli)	⋆67,72 (± 0,31)	⋆68,99 (± 0,40)	⋆79,62 (± 0,33)	⋆60,07 (± 0,87)	⋆62,94 (± 0,35)	⋆*68,25 (± 0,67)
Notre système (UFSAC, individuel, sans repli)	⋆71,08 (± 0,70)	⋆67,27 (± 0,65)	⋆82,79 (± 0,46)	⋆58,68 (± 1,38)	⋆66,67 (± 0,83)	⋆*72,57 (± 0,43)
Notre système (SemCor, ensemble, sans repli)	68,36	69,78	80,14	60,51	63,10	*68,63
Notre système (UFSAC, ensemble, sans repli)	72,54	69,57	83,05	59,63	67,47	*73,30
Notre système (SemCor, individuel, avec repli)	⋆73,18 (± 0,30)	⋆70,74 (± 0,40)	83,49 (± 0,32)	⋆61,54 (± 0,86)	67,55 (± 0,33)	⋆*72,24 (± 0,64)
Notre système (UFSAC, individuel, avec repli)	⋆72,30 (± 0,69)	⋆67,99 (± 0,65)	83,51 (± 0,46)	⋆58,84 (± 1,38)	68,07 (± 0,82)	⋆*73,34 (± 0,43)
Notre système (SemCor, ensemble avec repli)	73,71	**71,51**	**83,99**	**61,98**	67,70	*72,60
Notre système (UFSAC, ensemble avec repli)	**73,75**	70,27	83,77	59,78	**68,86**	***74,07**

Tableau 3. *Scores F1 (%) obtenus par notre système sur les tâches de DL des campagnes d'évaluation SensEval-SemEval. Les résultats préfixés par un astérisque (*) sont obtenus sur le corpus utilisé pour le développement pendant l'apprentissage. Les résultats préfixés par une étoile (⋆) sont significativement différents (entre le système entraîné sur le SemCor et le système entraîné sur tout UFSAC) selon un test de Student (p-valeur $< 0,01$). Les résultats en gras sont les meilleurs obtenus par notre système et par les systèmes de l'état de l'art. Ceux en rouge sont les meilleurs de l'état de l'art.*

Leur deuxième système (LSTM + LP) ajoute une étape de propagation de labels, dans laquelle ils augmentent automatiquement leurs données d'entraînement annotées en sens, en recherchant dans une grande quantité de textes non annotés des phrases similaires aux phrases annotées, et en portant les labels de sens depuis les phrases annotées, vers les phrases non annotées. Cette méthode apporte de meilleurs résultats sur la plupart des tâches, cependant ils récupèrent, pour leurs données non annotées, 1 000 phrases prises aléatoirement sur le Web pour chaque lemme, sans plus de précisions, ce qui rend la reproductibilité des résultats encore plus difficile.

Le système de Raganato *et al.* (2017) qui est, quant à lui, très semblable au nôtre obtient des résultats moins élevés alors qu'ils utilisent deux couches de cellules LSTM bidirectionnelles de taille 2 048 (1 024 par direction), donc un total de 4 096 unités cachées, ce qui est deux fois plus que notre modèle.

Pour leur second système (BLTM + att. + LEX + POS), les auteurs ont ajouté une couche d'attention à leur réseau, et ils effectuent de l'apprentissage multitâche, c'est-à-dire que leur réseau apprend à la fois à prédire un label de mot ou de sens, ainsi que la partie du discours (POS) du mot, et son label sémantique dans WordNet (LEX).

En comparaison avec ces autres systèmes, les nôtres obtiennent des scores supérieurs à ceux de Raganato *et al.* (2017) dans la majorité des cas, malgré une complexité réduite au niveau de l'architecture. Nous obtenons des scores similaires ou légèrement inférieurs à ceux de Yuan *et al.* (2016), mais en utilisant largement moins de données pour l'apprentissage, et surtout des données librement accessibles.

5.1. *Analyse des erreurs*

Afin de mieux comprendre l'impact du choix du corpus d'entraînement sur la qualité des annotations produites par notre système, nous proposons dans cette section une analyse plus fine des sorties, en nous appuyant sur les statistiques du tableau 4. Ces chiffres sont calculés avec les sorties de nos systèmes sur l'ensemble des corpus d'évaluation (sans le corpus de développement) et sans la stratégie de repli.

Tout d'abord la première chose que l'on peut remarquer, dans les deux premières sections du tableau, est que sur les 8 492 mots à annoter, le système appris sur l'ensemble des corpus UFSAC permet d'annoter 441 mots de plus que le système appris sur le SemCor, soit une couverture supplémentaire de plus de 5 % des mots à annoter. Sur les autres métriques, la précision du système appris sur plus de données est légèrement supérieure (moins de 1 %), et d'une manière générale le rappel et la F-mesure sont largement meilleurs (plus de 2 % de F1 du fait de la plus grande couverture).

La troisième section du tableau, « Moyenne du nombre de sens dans WordNet », montre une tendance visible sur les deux systèmes : les mots mal annotés sont généralement aussi les plus polysémiques. Cette statistique permet de rappeler que même pour les systèmes de désambiguïsation les plus performants, la distinction extrêmement « fine » entre les sens de WordNet reste un des défis principaux de cette tâche. Pour rappel, le verbe « *make* » a par exemple 49 sens différents dans WordNet.

	Notre système (SemCor)	Notre système (UFSAC)
Nombre de mots		
total des mots à annoter	8 492	8 492
mots non annotés	582	141
mots bien annotés	5 770	6 145
mots mal annotés	2 140	2 206
Métriques		
Couverture (C)	93,15 %	98,34 %
Précision (P)	72,95 %	73,58 %
Rappel (R)	67,95 %	72,36 %
F-mesure (F1)	70,36 %	72,97 %
Moyenne du nombre de sens dans WordNet		
pour les mots bien annotés	5,47	5,04
pour les mots mal annotés	8,58	8,86
Nombre moyen d'exemples du lemme cible dans le corpus d'entraînement		
pour les mots bien annotés	347	1 146
pour les mots mal annotés	215	1 489
Nombre moyen d'exemples du sens attendu dans le corpus d'entraînement		
pour les mots bien annotés	92	399
pour les mots mal annotés	21	206
Équilibre de la représentation des différents sens du lemme cible dans le corpus d'entraînement		
pour les mots bien annotés	1,64	2,00
pour les mots mal annotés	0,70	1,05
Nombre de mots mal annotés dont le sens attendu n'est jamais représenté dans le corpus d'entraînement	546	81

Tableau 4. *Analyse des erreurs commises sur l'ensemble des corpus d'évaluation (hormis le corpus de développement) par notre système entraîné sur le SemCor d'une part, et sur tous les corpus UFSAC d'autre part.*

Dans la quatrième section, « Nombre moyen d'exemples du lemme cible », on peut voir un indice supplémentaire pour montrer la qualité du SemCor par rapport aux autres corpus annotés en sens. En effet, la colonne de gauche montre que le système appris sur ce corpus annote généralement mieux les lemmes dont il a vu le plus d'exemples pendant l'entraînement, ce qui peut sembler évident pour un système supervisé. En revanche, le deuxième système affiche des chiffres opposés : les mots mal annotés ont été observés en moyenne un plus grand nombre de fois que les mots bien annotés. On peut ainsi émettre l'hypothèse que parfois, un trop grand nombre d'exemples dans le corpus d'entraînement va nuire à l'apprentissage des systèmes supervisés en ajoutant plus de bruit que d'informations pertinentes.

Le nombre moyen d'exemples du sens attendu dans le corpus d'entraînement montre que les sens les plus vus dans les données d'apprentissage sont généralement

plus souvent choisis par le système. Autrement dit si un sens est souvent représenté, que ce soit dans un contexte pertinent ou bien dans un contexte bruité, le système aura plus souvent tendance à le sélectionner.

L'avant-dernière section « Équilibre de la représentation des différents sens » met en avant une statistique particulière : pour les mots bien et mal annotés, est-ce que les différents sens du lemme cible étaient représentés de manière équilibrée dans le corpus d'entraînement ? Ces chiffres s'appuient sur la formule suivante :

$$\frac{nombre\ d'exemples\ du\ sens\ cible}{nombre\ d'exemples\ du\ lemme\ cible} * nombre\ de\ sens\ du\ lemme\ cible$$

Plus le chiffre est proche de 1, et plus la représentation du sens est « équilibrée », c'est-à-dire qu'il n'est pas sur-représenté ni sous-représenté par rapport aux autres sens possibles du lemme. Sur nos deux systèmes, les chiffres montrent qu'en général, les mots bien annotés ont en moyenne un sens qui est quasiment deux fois plus représenté que les autres sens, et les sens mal assignés sont largement moins représentés dans le cas du SemCor, et assez équilibrés dans le cas de l'ensemble de corpus.

Enfin, les derniers chiffres du tableau mettent en avant le fait que, dans de nombreux cas, sur ces corpus d'évaluation, même si le système est capable d'annoter un mot en sens, il ne pourra jamais sélectionner le sens attendu, tout simplement parce qu'il ne l'a jamais observé pendant l'apprentissage. C'est le cas pour 546 mots, pour le système appris sur le SemCor, ce qui correspond à plus de 6 % du total des mots à annoter, qu'aucun système supervisé appris sur ces données ne serait capable d'annoter correctement. Pour le deuxième système ce chiffre tombe à moins de 1 % du total des mots à annoter, cela montre une fois de plus l'intérêt d'utiliser l'ensemble des données disponibles pour l'entraînement de systèmes supervisés plus robustes.

Pour conclure, ces statistiques permettent de mieux se rendre compte de l'impact des corpus choisis pour l'entraînement sur les performances de notre système, mais aussi de l'importance de la qualité et de la quantité des données annotées en sens pour l'entraînement de systèmes supervisés en général.

En effet, si la majorité des systèmes supervisés sont entraînés sur le SemCor, très peu sont les travaux qui justifient son utilisation plutôt qu'un autre corpus. Avec l'ensemble des corpus annotés en sens WordNet maintenant facilement accessibles et dans un format unifié, il est désormais plus facile d'identifier et de sélectionner des corpus ou même seulement des parties de corpus qui seront bénéfiques aux systèmes de désambiguïsation supervisés.

6. Vers une amélioration non supervisée

Dans cette section, nous présentons une approche visant l'amélioration non supervisée de notre système, en s'appuyant sur des corpus non annotés. Nous mettons ainsi en avant des pistes qui pourraient être approfondies dans de futurs travaux.

6.1. *Approche*

L'approche que nous avons suivie est en partie inspirée de la méthode de propagation de labels de Yuan *et al.* (2016), dans laquelle les auteurs transfèrent des annotations de sens de leur corpus manuellement annoté vers des phrases non annotées, pour étendre leurs données d'apprentissage.

Notre approche est aussi inspirée des méthodes d'apprentissage par mimétisme telles que Buciluǎ *et al.* (2006), dans lesquelles un ou plusieurs modèles « enseignant » vont transférer leurs connaissances à un modèle « élève » en lui montrant comment effectuer une tâche. L'élève va ainsi apprendre à recopier ce que font les enseignants, observer des exemples dans de nouveaux contextes et ainsi apprendre à mieux généraliser.

Dans le contexte de la DL, les modèles enseignants sont des modèles capables d'annoter n'importe quelle séquence de mots en sens, et le modèle élève sera un nouveau modèle qui va être entraîné sur des données produites par les enseignants.

Intuitivement, on peut comprendre comment le modèle élève sera capable de généraliser les connaissances de l'enseignant grâce à l'exemple suivant : imaginons que les données d'entraînement utilisées par le système enseignant contiennent la phrase « la souris mange le fromage » annotée en sens, et que dans les données d'évaluation, on retrouve la phrase « la souris *dévore* le *gruyère* ». Les distances successives entre les mots « mange » et « *dévore* », puis entre « fromage » et « *gruyère* » dans l'espace vectoriel des mots sont peut être suffisamment élevées pour que le modèle ayant appris à annoter la première phrase ne soit pas capable d'annoter correctement la seconde.

Si par contre le modèle enseignant annote un ensemble de données dont la phrase « la souris mange le *gruyère* », ici seul le mot « *gruyère* » est différent par rapport à la phrase contenue dans les données d'entraînement et donc le mot « souris » a plus de chances d'être annoté correctement. Le modèle élève va ensuite pouvoir considérer cette dernière phrase annotée pendant son apprentissage, et par transfert, comme cette phrase est très proche de celle sur laquelle l'enseignant se trompait auparavant, il va cette fois être mieux capable de l'annoter correctement.

6.2. *Protocole expérimental*

Nous avons utilisé comme modèle enseignant le système de DL qui a obtenu le meilleur score F1 (voir le tableau 3) sur notre corpus de développement uniquement (SemEval 2015) afin d'éviter tout biais, c'est-à-dire celui entraîné sur toutes les données UFSAC avec la stratégie de repli.

Nous avons annoté avec ce système un million de phrases prises sur les données anglaises monolingues des campagnes d'évaluation de la traduction automatique WMT, et plus précisément le premier million de phrases du corpus « News Crawl 2016 » accessible sur le site de la campagne d'évaluation WMT17[7].

7. `http://data.statmt.org/wmt17/translation-task/news.2016.en.shuffled.gz`

Nous avons ensuite entraîné un nouveau modèle avec la même architecture sur ces données automatiquement annotées, en suivant le même protocole décrit dans la section 4, puis nous avons poursuivi l'entraînement de ce modèle, mais cette fois-ci sur les corpus UFSAC manuellement annotés, toujours pendant vingt cycles et en conservant le modèle avec le meilleur score sur le corpus de développement.

Nous avons réitéré cette dernière étape jusqu'à obtenir huit modèles différents afin d'évaluer cette méthode, comme pour le système original, à la fois en calculant la moyenne et l'écart-type des modèles pris individuellement, et en moyennant les prédictions d'un ensemble de modèles.

Dans le tableau 5 se trouvent les statistiques du corpus annoté en sens par le système enseignant, afin de le positionner par rapport aux autres corpus UFSAC. Tout d'abord nous pouvons voir que, comparativement à l'OMSTI qui est aussi un corpus d'environ un million de phrases, le nombre de mots de ce corpus est bien inférieur. En effet, en moyenne, les phrases de l'OMSTI sont constituées de 44 mots alors que les phrases de ce corpus ont une longueur moyenne de 22 mots. En ce qui concerne les nombres d'annotations en sens, ce nouveau corpus devient le plus densément annoté.

Concernant les lemmes distincts annotés, on retrouve plus de trois quarts du vocabulaire des corpus UFSAC qui est représenté dans ce million de phrases, et un très grand nombre d'exemples par lemmes et par sens, avec des chiffres comparables à ceux de l'OMSTI et du DSO.

Enfin, par rapport à la polysémie présente dans le corpus, et le ratio entre le nombre de sens présents dans le corpus par rapport au nombre de sens présents dans WordNet, on retrouve des chiffres similaires à ceux du SemCor.

	1M News 2016
Nombre de mots, en milliers	19 616
Nombre de mots annotés en sens, en milliers	7 762
Ratio entre le nombre de mots annotés et le nombre de mots total	0,40
Nombre de lemmes distincts annotés en sens	17 469
Nombre d'exemples par lemme	444
Nombre d'exemples par sens	181
Nombre de sens différents par lemme *dans WordNet*	3,26
Nombre de sens différents par lemme *dans le corpus*	1,60
Ratio entre le nombre de sens différents par lemme dans le corpus et le nombre de sens différents par lemme dans WordNet	0,53

Tableau 5. *Statistiques du corpus constitué du premier million de phrases du corpus « News Crawl 2016 » annoté automatiquement en sens par le système enseignant.*

6.3. *Résultats*

Nous avons évalué le système « élève » avec repli sur les mêmes tâches que précédemment, et nous avons comparé ses scores avec ceux obtenus par le système « enseignant », et avec le système de Yuan *et al.* (2016). Les résultats sont dans la tableau 6.

Système	SE2	SE3	SE07 (07)	SE07 (17)	SE13 (12)	SE15 (13)
Yuan *et al.* (2016) (LSTM)	73,6	69,2	82,8	64,2	67,0	72,1
Yuan *et al.* (2016) (LSTM + LP)	73,8	71,8	83,6	63,5	69,5	72,6
Système « enseignant » (UFSAC, ensemble)	73,75	**70,27**	83,77	59,78	68,86	*74,07
Système « élève » à l'initialisation (1M News 2016, individuel)	71,25	68,38	82,93	61,10	68,80	*71,53
Système « élève » (UFSAC + 1M News 2016, individuel)	73,50 (± 0,68)	68,74 (± 0,66)	84,39 (± 0,50)	59,62 (± 0,95)	68,95 (± 0,81)	***74,43** (± 0,57)
Système « élève » (UFSAC + 1M News 2016, ensemble)	**74,58** (+ 0,83)	69,35 (-0,92)	**84,96** (+ 1,19)	**60,66** (+ 0,88)	**69,77** (+ 0,91)	*74,17 (+ 0,10)

Tableau 6. *Scores F1 (%) obtenus par le système élève sur les tâches de DL des campagnes d'évaluation SensEval-SemEval. Les résultats préfixés par un astérisque (*) sont obtenus sur le corpus utilisé pour le développement pendant l'apprentissage. La différence entre le meilleur système élève et le système enseignant est affichée entre parenthèses. Le meilleur score entre l'élève et l'enseignant est affiché en gras. Le meilleur score de l'état de l'art est affiché en rouge.*

Sur toutes les tâches d'évaluation, à part celle de SensEval 3, le système élève obtient ainsi des scores significativement supérieurs à ceux du système enseignant, et il obtient même des scores surpassant l'état de l'art sur les tâches de SensEval 2, SemEval 2007 (07), SemEval 2013 et SemEval 2015.

À travers ces résultats, on voit à quel point la mise en place de ce type d'apprentissage par transfert de connaissances peut s'avérer efficace pour la construction d'un système de DL robuste et de bonne qualité. Notre système ainsi entraîné obtient en effet des scores supérieurs à notre système original et à l'état de l'art sur la plupart des tâches d'évaluation, alors que nous avons uniquement utilisé comme ressource supplémentaire des phrases en anglais non annotées provenant d'un corpus libre d'accès.

Afin d'avoir plus de détails sur les performances de notre système « élève », nous proposons dans le tableau 7 une analyse d'erreurs semblable à celle du tableau 4. Ces statistiques sont similaires à celles du système enseignant (voir tableau 4), mais les tendances sont plus marquées. L'écart se creuse entre le nombre d'exemples du sens attendu pour les mots bien annotés et pour les mots mal annotés, ainsi que pour l'équilibre de la représentation des sens. Une tendance change cependant, c'est le nombre d'exemples du lemme cible pour les mots bien annotés et mal annotés. Dans tous ces cas, on a maintenant une bien plus forte proportion d'exemples de lemmes et de sens des mots bien annotés par rapport à ceux mal annotés.

	Système « élève »
Nombre de mots	
total des mots à annoter	8 492
mots non annotés	141
mots bien annotés	6 188
mots mal annotés	2 163
Métriques	
Couverture (C)	98,34 %
Précision (P)	74,10 %
Rappel (R)	72,87 %
F-mesure (F1)	73,48 %
Moyenne du nombre de sens dans WordNet	
pour les mots bien annotés	5,08
pour les mots mal annotés	8,82
Nombre moyen d'exemples du lemme cible dans le corpus d'entraînement	
pour les mots bien annotés	13 743
pour les mots mal annotés	11 893
Nombre moyen d'exemples du sens attendu dans le corpus d'entraînement	
pour les mots bien annotés	4 311
pour les mots mal annotés	1 255
Équilibre de la représentation des différents sens du lemme cible dans le corpus d'entraînement	
pour les mots bien annotés	2,32
pour les mots mal annotés	0,96
Nombre de mots mal annotés dont le sens attendu n'est jamais représenté dans le corpus d'entraînement	75

Tableau 7. *Analyse des erreurs commises sur l'ensemble des corpus d'évaluation (hormis le corpus de développement) par notre système « élève » pré-entraîné sur notre corpus automatiquement annoté et entraîné sur tous les corpus UFSAC.*

Cette approche est un premier pas pour l'amélioration d'un système de DL comme le nôtre, sans utiliser de données annotées manuellement supplémentaires. Elle l'aide effectivement à mieux généraliser, mais elle souffre encore de défauts, en témoigne la baisse de résultats sur la tâche de SensEval 3. Afin de l'améliorer, nous prévoyons de sélectionner plus finement les données à annoter par le système enseignant, afin de s'adapter à la tâche sur laquelle on souhaite s'évaluer, et de sélectionner les annotations produites par le système enseignant, pour éviter de reproduire les erreurs du modèle neuronal qui peuvent être facilement détectées, par exemple à l'aide d'une mesure de confiance s'appuyant sur sa couche de sortie. Enfin, des architectures neuronales plus évoluées, comme par exemple les réseaux récurrents à base d'arbres (Tai *et al.*, 2015), pourraient permettre d'améliorer notre système en tenant mieux compte de la structure syntaxique des phrases en entrée.

7. Conclusion

Nous présentons dans cet article une nouvelle architecture de réseau neuronal pour la désambiguïsation lexicale à base de cellules LSTM. Ces cellules récurrentes sont largement utilisées dans les réseaux de neurones traitant des séquences tels que les systèmes *sequence-to-sequence* pour la traduction automatique ou les systèmes utilisant un modèle de langue prédisant la prochaine entrée d'une suite de mots. Notre modèle est composé d'une couche d'entrée qui prend une séquence de vecteurs de mots construits séparément, puis une couche cachée de cellules LSTM bidirectionnelles, et enfin une couche de sortie entièrement connectée de la taille du nombre de sens possibles dans le dictionnaire utilisé.

Nous avons entraîné un système sur six corpus au format UFSAC, à savoir le SemCor, le DSO, le WNGT, l'OMSTI, le MASC et l'OntoNotes, mais aussi un système sur le SemCor uniquement, et nous les avons évalués sur les tâches de DL des campagnes d'évaluation SensEval-SemEval. Les résultats montrent que nos systèmes obtiennent des scores équivalents à ceux des meilleurs systèmes neuronaux de l'état de l'art. Seul le système de Yuan *et al.* (2016) augmenté par les données issues de leur propagation de labels obtient des scores plus élevés. Cette augmentation indépendante de leur architecture neuronale s'appuie cependant sur l'utilisation de grandes quantités de textes pris aléatoirement sur le Web, ce qui rend la reproductibilité difficile.

Nous avons présenté une analyse fine des résultats de nos deux systèmes afin de comprendre l'impact du choix des corpus d'entraînement pour l'apprentissage. Nous montrons ainsi les effets positifs d'utiliser davantage de données annotées en sens, mais aussi les problèmes que peuvent apporter des données de moins bonne qualité. Nous espérons ainsi voir se développer des approches supervisées utilisant d'autres corpus que le SemCor.

Enfin, nous avons présenté une amélioration de notre système à l'aide d'une approche par transfert de connaissances pour laquelle seulement un million de phrases initialement non annotées étaient ajoutées aux données d'entraînement afin d'obtenir un modèle plus robuste et performant. Nous avons présenté des résultats avec ce système qui surpassent significativement l'état de l'art sur la plupart des tâches d'évaluation de la DL, et nous avons proposé quelques pistes d'amélioration futures pour continuer dans cette voie.

Les études sur les réseaux de neurones pour la DL sont encore très récentes comme en atteste le faible nombre de systèmes existants pour le moment. C'est cependant une direction prometteuse, tant les résultats obtenus par ces nouveaux systèmes ont montré leur qualité sur les campagnes d'évaluation.

8. Bibliographie

Brody S., Lapata M., « Good Neighbors Make Good Senses : Exploiting Distributional Similarity for Unsupervised WSD », *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, Manchester, UK, p. 65-72, 2008.

Buciluǎ C., Caruana R., Niculescu-Mizil A., « Model compression », *Proceedings of the 12th international conference on Knowledge discovery and data mining*, p. 535-541, 2006.

Chan Y. S., Ng H. T., Chiang D., « Word Sense Disambiguation Improves Statistical Machine Translation », *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, p. 33-40, 2007a.

Chan Y. S., Ng H. T., Zhong Z., « NUS-PT : Exploiting Parallel Texts for Word Sense Disambiguation in the English All-words Tasks », *Proceedings of the 4th International Workshop on Semantic Evaluations*, Association for Computational Linguistics, p. 253-256, 2007b.

Decadt B., Hoste V., Daelemans W., Bosch A. V. d., « GAMBL, genetic algorithm optimization of memory-based WSD », *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, 2004.

Edmonds P., Cotton S., « SENSEVAL-2 : Overview », *The Proceedings of the Second International Workshop on Evaluating Word Sense Disambiguation Systems*, SENSEVAL '01, Association for Computational Linguistics, Stroudsburg, PA, USA, p. 1-5, 2001.

Hochreiter S., Schmidhuber J., « Long Short-Term Memory », *Neural Computation*, vol. 9, n° 8, p. 1735-1780, 1997.

Hoste V., Kool A., Daelemans W., « Classifier Optimization and Combination in the English All Words Task », *The Proceedings of the Second International Workshop on Evaluating Word Sense Disambiguation Systems*, p. 83-86, 2001.

Hovy E., Marcus M., Palmer M., Ramshaw L., Weischedel R., « OntoNotes : The 90% Solution », *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume : Short Papers*, p. 57-60, 2006.

Iacobacci I., Pilehvar M. T., Navigli R., « Embeddings for Word Sense Disambiguation : An Evaluation Study », *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 897-907, 2016.

Ide N., Baker C., Fellbaum C., Fillmore C., Passonneau R., « MASC : the Manually Annotated Sub-Corpus of American English », *Proceedings of the Sixth International Conference on Language Resources and Evaluation*, 2008.

Kågebäck M., Salomonsson H., « Word Sense Disambiguation using a Bidirectional LSTM », *5th Workshop on Cognitive Aspects of the Lexicon (CogALex)*, 2016.

Kilgarriff A., « SENSEVAL : An Exercise in Evaluating Word Sense Disambiguation Programs », *First international conference on language resources and evaluation*, 1998.

Kingma D. P., Ba J., « Adam : A Method for Stochastic Optimization », *Proceedings of the 3rd International Conference for Learning Representations*, 2015.

Lesk M., « Automatic sense disambiguation using MRD : how to tell a pine cone from an ice cream cone », *Proceedings of SIGDOC '86*, ACM, New York, NY, USA, p. 24-26, 1986.

Liu F., Lu H., Neubig G., « Handling Homographs in Neural Machine Translation », *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, p. 1336-1345, 2018.

Mikolov T., Sutskever I., Chen K., Corrado G. S., Dean J., « Distributed Representations of Words and Phrases and their Compositionality », *Advances in Neural Information Processing Systems 26*, p. 3111-3119, 2013.

Miller G. A., « WordNet : a lexical database for English », *Communications of the ACM*, vol. 38, n° 11, p. 39-41, 1995.

Miller G. A., Leacock C., Tengi R., Bunker R. T., « A semantic concordance », *Proceedings of the workshop on Human Language Technology*, HLT '93, Association for Computational Linguistics, Stroudsburg, PA, USA, p. 303-308, 1993.

Moro A., Navigli R., « SemEval-2015 Task 13 : Multilingual All-Words Sense Disambiguation and Entity Linking », *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, p. 288-297, 2015.

Navigli R., « WSD : a Survey », *ACM Computing Surveys*, vol. 41, n° 2, p. 1-69, 2009.

Navigli R., Jurgens D., Vannella D., « SemEval-2013 Task 12 : Multilingual Word Sense Disambiguation », *Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, p. 222-231, 2013.

Navigli R., Litkowski K. C., Hargraves O., « SemEval-2007 Task 07 : Coarse-Grained English All-Words Task », *SemEval-2007*, Prague, Czech Republic, p. 30-35, June, 2007.

Navigli R., Ponzetto S. P., « BabelNet : Building a very large multilingual semantic network », *48th annual meeting of the association for computational linguistics*, p. 216-225, 2010.

Ng H. T., Lee H. B., « DSO Corpus of Sense-Tagged English », 1997.

Pennington J., Socher R., Manning C. D., « GloVe : Global Vectors for Word Representation », *Empirical Methods in Natural Language Processing (EMNLP)*, p. 1532-1543, 2014.

Pradhan S. S., Loper E., Dligach D., Palmer M., « SemEval-2007 Task 17 : English Lexical Sample, SRL and All Words », *Proceedings of the 4th International Workshop on Semantic Evaluations*, p. 87-92, 2007.

Raganato A., Delli Bovi C., Navigli R., « Neural Sequence Learning Models for Word Sense Disambiguation », *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, p. 1167-1178, 2017.

Rios A., Mascarell L., Sennrich R., « Improving Word Sense Disambiguation in Neural Machine Translation with Sense Embeddings », *Proceedings of the Second Conference on Machine Translation, Volume 1 : Research Papers*, Copenhagen, Denmark, 2017.

Snyder B., Palmer M., « The English all-words task », *Proceedings of the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, 2004.

Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R., « Dropout : A Simple Way to Prevent Neural Networks from Overfitting », *J. Mach. Learn. Res.*, vol. 15, n° 1, p. 1929-1958, 2014.

Sutskever I., Vinyals O., Le Q. V., « Sequence to Sequence Learning with Neural Networks », *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, MIT Press, Cambridge, MA, USA, p. 3104-3112, 2014.

Taghipour K., Ng H. T., « One Million Sense-Tagged Instances for Word Sense Disambiguation and Induction », *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, Association for Computational Linguistics, p. 338-344, 2015.

Tai K. S., Socher R., Manning C. D., « Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks », *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1 : Long Papers)*, p. 1556-1566, 2015.

Vial L., Lecouteux B., Schwab D., « UFSAC : Unification of Sense Annotated Corpora and Tools », *Language Resources and Evaluation Conference (LREC)*, 2018.

Vial L., Tchechmedjiev A., Schwab D., « Extension lexicale de définitions grâce à des corpus annotés en sens », *23ème Conférence sur le Traitement Automatique des Langues Naturelles*, 2016.

Yarowsky D., « Unsupervised Word Sense Disambiguation Rivaling Supervised Methods », *33rd Annual Meeting on Association for Computational Linguistics*, p. 189-196, 1995.

Yuan D., Richardson J., Doherty R., Evans C., Altendorf E., « Semi-supervised Word Sense Disambiguation with Neural Models », *COLING 2016*, 2016.

Zhong Z., Ng H. T., « It Makes Sense : A Wide-coverage Word Sense Disambiguation System for Free Text », *Proceedings of the ACL 2010 System Demonstrations*, p. 78-83, 2010.

Zhong Z., Ng H. T., « Word Sense Disambiguation Improves Information Retrieval », *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 273-282, 2012.

Notes de lecture

Rubrique préparée par Denis Maurel

Université de Tours, LIFAT (Laboratoire d'informatique fondamentale et appliquée)

Ruslan MITKOV, Johanna MONTI, Gloria Corpas PASTOR, Violeta SERETAN. Multiword Units in Machine Translation and Translation Technology. *John Benjamins publishing company*. 2018. 259 pages. ISBN 978-90-272-0060-0.

Lu par **Christian BOITET**

Université de Grenoble-Alpes / GETALP

Multiword Units in Machine Translation and Translation Technology *est un volume de la collection* Current issues in linguistic theory *de John Benjamins. Les références s'arrêtent à 2013, 2014 ou 2015 selon les chapitres du volume. Du coup, il n'y a rien sur la TA neuronale, ce qui est dommage pour un livre sorti en 2018. Il y a d'abord un très long chapitre introductif (trente-sept pages dont douze de bibliographie), dont le but est de présenter le domaine, puis les trois parties, concernant les EML*[1] *en TA, les EML dans des applications multilingues du TAL et l'identification et la traduction des EML. Cette répartition serait à revoir. Par exemple, le dernier article porte sur la traduction des EML dans le système FIPS-2 et serait bien mieux placé dans la partie 1.*

Une remarque préliminaire importante est que, semble-t-il, tous les auteurs, à l'exception notable d'Éric Wehrli et Luka Nerima (dernier article), ignorent les contributions essentielles d'Igor Mel'tchuk à la représentation et au traitement des EML, qui ont commencé… en 1958, avec l'introduction des fonctions lexico-sémantiques (FLS), puis un grand développement de la théorie, l'incorporation dans plusieurs dictionnaires explicatifs et combinatoires (DEC), dans des systèmes de TA, et un très grand nombre de publications, en russe, en français et en anglais (au moins). Dès 1961, le CETA à Grenoble, dirigé par Bernard Vauquois, utilisait certaines FLS (dérivationnelles et productives) dans son système de TA russe français ! De ce point de vue, ce volume est assez décevant. En effet, les FLS sont une vraie clé pour la représentation et le traitement des collocations de type lexico-sémantique. Cependant, on trouve des choses assez intéressantes dans ce volume. Parcourons-le.

1 J'utilise en français le terme EML (expressions multilexicales), de préférence à d'autres termes.

Introduction

La longue introduction du volume, rédigée par les éditeurs, commence, bien sûr, par un petit historique des études sur les EML. On passe directement de Firth (1957) à Sag (2002)… comme si rien ne s'était passé en quarante-cinq ans. Or, il n'y a pas eu que Mel'tchuk à s'y intéresser durant cette période ! On a mentionné le CETA, mais il faut dire que pas mal d'autres systèmes de TA ont abordé le problème et trouvé des solutions pour beaucoup de types d'EML. Peter Toma a introduit dans Systran un « dictionnaire d'expressions » dès 1967, Slocum traitait les mots composés allemands dans METAL dès 1982, etc., et il y a dans cette introduction une section (3.2) qui décrit le traitement des EML dans OpenLogos.

On note aussi des affirmations bien en retard sur l'état de l'art : au 2.2, on nous dit que la recherche à venir dans ce domaine va surmonter les limitations de l'approche dominante « mots-avec-des-blancs ». Mais enfin, la thèse de Violeta Seretan (une des éditrices !) en 2008 a très largement prouvé et surmonté ces limitations, en montrant qu'une bien meilleure approche consiste à travailler sur des arbres linguistiques… C'était donc dix ans avant la sortie de ce volume. On regrette aussi qu'il n'y ait pas d'article sur le projet PARSEME, qui a produit des résultats vraiment très intéressants sur la détection et le traitement des EML en contexte multilingue depuis 2015.

Le côté positif de cette introduction est sa très grande richesse en références bibliographiques et autres pointeurs (projets, systèmes), en ce qui concerne la période 2000-2015 environ. Passons aux différentes parties.

Partie 1

L'étude du chapitre 1 concerne essentiellement les constructions du type verbe + nom en espagnol. Il y a une petite évaluation de la qualité des résultats de la traduction des quatre-vingt-dix-neuf exemples par le système enrichi avec le traitement des MLE. Dommage qu'il n'y ait aucune précision sur ce traitement (ni de référence au site Apertium, où on devrait pouvoir le trouver).

Le chapitre 2 n'est pas vraiment dans le thème de cette partie et commence mal, avec l'affirmation fausse et non justifiée que la SMT est meilleure que la RBMT pour traduire les EML. Plusieurs articles de ce volume prouvent que c'est le contraire. Les auteurs prétendent que les systèmes SMT (de type Moses ici) peuvent extraire les EML des corpus et donc avoir une meilleure couverture que les systèmes RBMT (qu'on devrait plutôt appeler systèmes fondés sur la connaissance linguistique, ou systèmes « experts », que systèmes à règles, passons sur cette confusion assez généralement répandue). C'est tout à fait faux, car bien des systèmes de TA « experts » incluent des dictionnaires de tournures, idiomes, termes techniques, etc. tout simplement « récupérés » de dictionnaires informatisés et de terminologies en ligne (comme iate.eu). À titre indicatif, ATLAS-II de Fujitsu avait, en 2009 environ, 6,5 M entrées dans ses dictionnaires, Systran entre 600 K et 800 K par langue, etc.). C'est beaucoup plus que ce que l'on trouve dans les tables de

traductions calculées par Moses sur, par exemple, les 20 M de phrases du corpus de l'EU (plus qu'EuroParl).

À part cela, l'étude avec des étudiants post-éditant des résultats de TA est intéressante, mais elle n'apporte rien à l'aspect TAL des EML.

Le chapitre 3 cherche à traiter les EML dans le cadre de la SMT à la Moses, sans utiliser l'approche hiérarchique qui seule permet de travailler sur des arbres (mieux, sur des bi-arbres). Donc les collocations sont toujours traitées dans le cadre d'une fenêtre de quelques mots (mots-formes), ce qui interdit par exemple de traiter convenablement les verbes à particule séparable et séparée de l'allemand et même de l'anglais. Quelques expériences sont présentées, mais la « mesure de qualité » est BLEU ou BLEU 2, et tout le monde devrait savoir, depuis le fameux article de Osborne, Callison-Burch et Koehn, que BLEU ne mesure PAS la qualité (que ce soit la qualité linguistique ou la qualité d'usage), et, pour plusieurs raisons théoriques, ne peut pas la mesurer.

Partie 2

Le chapitre 4 est vraiment très intéressant et à lire en détail. L'auteur a une très longue expérience du TAL pour la TA, depuis ses débuts à Siemens sur le système METAL, puis à Sietech (une filiale de Siemens), puis à GMS, Lernout&Hauspie, Sail Labs (toujours sur METAL), puis sur la version de LMT (IBM, McCord) améliorée par Linguatec (Heidelberg et Munich) en PMT, puis chez GMX… Le contexte est celui de la RI vers l'allemand, à partir de langues européennes, et de langues présentant une grande variabilité au niveau de l'usage des espaces (arabe, persan, pashto, turc). Il s'agit d'extraction d'informations multilingues (des mots-clés et des entités nommées), et pas de TA proprement dite (quoique les réponses doivent être fournies en allemand, mais ce ne sont pas des phrases). Cependant, ce sont bien les techniques efficaces en TA d'énoncés qui sont recherchées et présentées.

L'article est à la fois très érudit et très technique, avec des références précises aux niveaux X0, X1, X2 de la théorie X-barre. Les conclusions sont très intéressantes, et effectivement démontrées. (1) Il est important (et possible) de traiter de façon uniforme les unités lexicales, qu'elles soient simples ou complexes. (2) Le défi du traitement des EML se situe essentiellement en analyse et est monolingue, pas bilingue, le transfert lexical devant être le même, qu'il s'agisse d'unités simples ou complexes. (3) Dans les dictionnaires monolingues, les EML doivent avoir tous les attributs (traits) des unités lexicales (UL) simples, plus la spécification de la tête de l'EML et la liste des UL composantes, avec leur lemme et leur partie du discours (POS). (4) Il ne faut pas traiter les EML en prétraitement (approche « mots-avec-des-blancs »), ni après l'analyse (trop tard !), mais comme une partie intégrante de l'analyse. Du point de vue linguistique, l'approche consiste à ajouter un niveau au schéma X-barre, niveau où on appliquera les règles relatives aux EML. (5) Enfin, la décision du choix d'utiliser un sens non compositionnel (EML) ou un

sens compositionnel (synthétique) dépend du contexte et peut en pratique être fondée sur des scores (probabilistes, possibilistes ou préférentiels).

Le chapitre 5 pose le problème de la terminologie : sous prétexte que les mots composés allemands n'ont pas de séparateurs internes, on ne les considère pas comme de type EML. C'est d'autant plus injustifié qu'en allemand la même unité, un verbe à particule séparable, peut se présenter en deux mots (typographiques) séparés (« *er kam am Morgen zurück* ») ou collés (« *er ist am Morgen zurückgekommen* »).

Cela dit, le contenu de ce chapitre est intéressant. On utilise une convention pour noter la segmentation d'un mot composé en ses parties lexicales et dérivationnelles : « Brauch~er#schutz#dienst#list~ung » (service de protection du consommateur). Dommage que l'on n'aille pas jusqu'à une description hiérarchique (nécessaire pour la synthèse vocale d'ailleurs), comme : « [[Haupt#[bahn#hof]]#[[ge~päck]#[auf~be[#wahr~ung]]]] » (consigne à bagages de la gare principale).

Ces alignements sont intéressants car ils montrent que, comme dans les énoncés complets, ils ne sont pas un à un. Par exemple, « Korruption~s#be~kämpf~ung#ein~heit » est en français « unité anti-corruption » (plutôt que « unité de lutte contre la corruption », mais « *órgano contra la corrupción* » (« *contra* » vient de ε et « *bekämpfung* » va à ε). Au total, c'est une étude intéressante, mais pas directement reliée au thème de cette partie, ni d'ailleurs à celui de la TA.

Le but du chapitre 6 est ici d'obtenir de meilleurs alignements que quand on aligne des mots formes (mots typographiques). Le processus de segmentation est proche de celui de l'allemand décrit dans le chapitre précédent. Une amélioration est obtenue quand on aligne les « sous-mots » néerlandais avec les mots (typographiques) de l'autre langue (ici, anglais pour un corpus médical, français pour un corpus automobile). Le point intéressant est qu'une amélioration plus importante est obtenue en appliquant aux deux alignements obtenus une heuristique d'intersection, puis en fusionnant tous les points d'alignement, et enfin en ajoutant des points d'alignement venant du résultat du module grow-diag-final dans le cas du modèle d'alignement entraîné sur les alignements des sous-mots. Les F-mesures relatives à l'extraction de terminologies bilingues restent cependant assez faibles (les meilleures à 56 % ou 54 %). Cela nous renforce dans l'idée que, pour obtenir de bons résultats, il faut passer des alignements entre chaînes à des alignements entre arbres.

Partie 3

Le chapitre 7 est assez court, et n'apporte pas grand-chose de nouveau. Surtout, on voit bien que les règles de correspondance font intervenir un modèle d'alignement de chaînes, et pas d'arbres, alors que la thèse de V. Seretan datait de 2008, six ans avant la rédaction de cet article. Il y a bien un article de Seretan de 2011 cité dans le texte, mais pas pour le fond, seulement pour étayer le fait que

les collocations sont la majorité des ELM. Dans le même ordre d'idée, le chapitre devrait tout de même discuter les résultats présentés par Estelle Delpech, en 2013, dans sa thèse thèse (prix de l'ATALA). Elle y montre, en résumé, que l'extraction de terminologie technique bilingue (de qualité) à partir de corpus comparables ne fonctionne pas et ne peut pas fonctionner…

Le chapitre 8, une étude, assez spéculative sur « la possibilité d'utiliser des faisceaux lexicaux bilingues pour améliorer le degré de naturel et de fidélité (*textual fit*, littéralité ?) des textes traduits [automatiquement, par des systèmes Moses] ». Les faisceaux lexicaux sont des séquences de trois à sept mots ayant des fonctions discursives similaires, apparaissant dans un corpus comparable anglais-polonais de tracts d'information à destination de patients. À cause des différences translinguistiques, on applique de plus un certain nombre de critères formels pour filtrer les faisceaux dans chaque sous-corpus. Les résultats montrent que les faisceaux lexicaux bilingues extraits de corpus comparables ont un potentiel inexploré pour la TA, les aides à la traduction, et la lexicographie bilingue. Au total, on voit mal comment on pourrait, comme l'affirme l'auteur, intégrer les résultats de cette étude dans des systèmes de TA et de THAM (aide à la traduction).

Le chapitre 9 est une description très précise, intéressante et illustrée de la représentation et du traitement des tournures en croate à l'aide de NooJ. La représentation correspond à la partie « haute » des transducteurs, et les actions (balisage, traduction…) à la partie basse des transitions. Comme NooJ permet de traiter non seulement des transducteurs classiques (réguliers), mais aussi des réseaux de transition récursifs (équivalents aux grammaires hors contexte, mais plus puissants quand on utilise des contextes) permettant de travailler, en fait, sur des arbres, on a la possibilité de traiter des ELM non connexes.

Du point de vue de la traduction, ou d'autres applications multilingues, rien n'est dit. La bibliographie est assez courte, et on s'étonne de ne pas y trouver en bonne place au moins une référence à l'article de Šandor Dembitz *et al.* qui décrit une énorme collection allant jusqu'à des octogrammes, qui doivent contenir beaucoup de tournures.

Le chapitre 10 est une étude très détaillée, certes, mais on ne voit pas ce que cela apporte à la TA ou à la THAM. En fait, la méthode utilisée montre en un sens ce qu'il ne faut pas faire ! Ici, par exemple, on prend le verbe « faire » et on cherche à détailler tous ses sens possibles. Mais (et c'est dit dans le texte), son sens dépend le plus souvent du lexème en collocation à sa droite. Cela veut simplement dire que, le plus souvent, ce verbe est un verbe support.

L'étude linguistique présentée ici gagnerait donc à être structurée autour du ou des « sens propres» (comme créer, agir…), des FLS dont doći peut être une valeur, et de son ou ses rôles et sens possibles dans des idiomes (comme « faire un tabac », « faire le beau », « faire fort », etc.).

Le chapitre 11 est de loin le chapitre le plus intéressant et le plus avancé du volume. L'identification des collocations, la résolution des anaphores et la

traduction sont connues comme des problèmes parmi les plus difficiles du TAL. Ici, ils sont traités ensemble, et avec succès.

Comme dit plus haut, la meilleure méthode d'identification des collocations dans des corpus est due à É. Wehrli et son équipe. Ici, il s'agit de les identifier quand des éléments en sont élidés ou pronominalisés. Exemple : « *Paul broke the word record. He broke it by a large margin* ». Même si un système reconnaît les deux EML (*break* est verbe support de record, et large ∈ Magn(*margin*)), il produit actuellement au mieux : « Paul a battu le record du monde. Il l'a brisé avec un grand écart ». La solution présentée, implémentée dans l'analyseur Fips-5, consiste à faire évoluer une mémoire des centres de reprises anaphoriques possibles, et à la transmettre d'un énoncé au suivant au cours de l'analyse. On arrive alors à améliorer la traduction, et à produire : « Il l'a battu avec une grande marge ».

Ne serait-ce que pour ce dernier article, il est donc tout à fait intéressant pour un taliste de lire ce volume, tout du moins les chapitres signalés ici comme les plus intéressants.

Pierre M. NUGUES. Language Processing with Perl and Prolog. Theories, Implementation, and Application. *Springer*. 2014. 662 pages. ISBN 978-3-642-41463-3.

Lu par **Caroline BARRIERE**

Université d'Ottawa, École de science informatique et génie électrique

---

De prime abord, le livre de Nugues est sans contredit « intimidant ». C'est d'abord son titre qui m'a intimidée. Pourquoi Perl ou encore Prolog dans une ère où le Java ou encore plus le Python prévalent? C'est ensuite sa taille qui m'a intimidée, me retrouvant devant un volume de 650 pages sans savoir trop par où débuter.

Alors, j'ai débuté, par le début…. Le premier chapitre « *An Overview of Language Processing* » est vraiment un tour de force en tant que résumé de divers aspects du traitement automatique des langues (TAL). J'ai beaucoup aimé. De façon concise et claire, Nugues nous parle autant de syntaxe que de sémantique et de dialogues. Je recommande la lecture de ce premier chapitre à toute personne désirant avoir un aperçu du TAL.

Poursuivant ma lecture, j'ai constaté que, de façon soutenue, l'auteur écrit bien, de façon claire et précise, et qu'il structure ses chapitres de façon uniforme, présentant d'abord les concepts théoriques, suivis d'exemples pratiques, parfois d'implémentation en Prolog, terminant par une section de « *further reading* »et quelques exercices. J'apprécie cette cohérence.

Quant à l'ordre de lecture, j'ai rapidement abandonné un ordre linéaire. Même si l'auteur nous dit qu'il s'agit d'un livre de cours, j'ai plutôt trouvé que la structure du livre ne suivait pas tant un ordre naturel pour l'enseignement du TAL, mais couvrait

plutôt beaucoup de sujets différents, tels qu'on le verrait dans une encyclopédie. Un avantage de cette organisation encyclopédique est que nous pouvons lire les chapitres qui nous intéressent dans l'ordre que nous désirons.

Ma déception a été le manque de références vers des articles récents et des applications récentes. Il est dommage que cette deuxième édition de *Language Processing with Perl and Prolog*, datant de 2014 (la première édition datant de 2006), pointe encore vers plusieurs applications des années 90, voire même des années 80. Lors de la lecture des sections « *Further Reading* », ou même la lecture des sections applicatives, le manque de références récentes laisse un arrière-goût de désuétude. C'est d'autant plus dommage que l'auteur sait bien présenter et expliquer les concepts théoriques de tout temps, parlant autant d'automates à états finis, que de réseaux de neurones, que de modèles markoviens, ou que de grammaire de Chomsky. Non seulement la couverture théorique est large en termes de sujets et d'époques (des années 1970 à 2010), mais en plus, pour chaque sujet, Nugues sait donner juste assez d'informations pour la compréhension des principes sans se perdre dans les détails.

Comme tout livre en TAL, le champ est tellement vaste que l'auteur y injecte ses préférences… Le livre de Nugues a un biais vers la syntaxe. Sur l'ensemble des dix-sept chapitres du livre, on retrouve sept chapitres touchant l'analyse syntaxique Alors, les amoureux de syntaxe, c'est un livre pour vous!

Quant aux langages de programmation suggérés dans le titre du livre, Perl et Prolog, j'ai rapidement constaté que Perl était relégué à l'arrière-plan, n'étant présenté que dans une section discutant des expressions régulières, mais qu'en contrepartie, Prolog était omniprésent. En effet, presque chaque chapitre contient une section sur l'implémentation en Prolog des concepts présentés. De plus, une annexe « *An Introduction to Prolog* », de cinquante pages, pourrait faire en soit le contenu d'un excellent tutoriel sur l'utilisation de Prolog pour le TAL.

En conclusion, je pense que malgré mon intimidation initiale, j'ai su apprécier la clarté et la concision dont fait preuve Nugues sur les divers sujets. Je suggérerais aux lecteurs hésitants de ne pas se laisser dérouter par le titre mentionnant le Perl et le Prolog, et de profiter du contenu de ce livre plutôt comme un manuel de référence sur les concepts en TAL, que ce soit pour découvrir la sémantique, le dialogue, l'analyse de corpus, ou encore, plus certainement, l'analyse syntaxique.

Résumés de thèses

Rubrique préparée par Sylvain Pogodalla

Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
sylvain.pogodalla@inria.fr

Rachel BAWDEN : rachel.bawden@ed.ac.uk

Titre : Au-delà de la phrase : traduction automatique de dialogue en contexte

Mots-clés : Traduction automatique, contexte, dialogue, discours.

Title: *Going beyond the Sentence: Contextual Machine Translation of Dialogue*

Keywords: *Machine translation, context, dialogue, discourse.*

Thèse de doctorat en informatique, LIMSI, CNRS, Université Paris-Sud, Université Paris-Saclay, Orsay, sous la direction de Sophie Rosset (DR, CNRS, LIMSI) et Thomas Lavergne (MC, Université Paris-Sud, Université Paris-Saclay, LIMSI). Thèse soutenue le 29/11/2018.

Jury : Mme Sophie Rosset (DR, CNRS, LIMSI, codirectrice), M. Thomas Lavergne (MC, Université Paris-Sud, Université Paris-Saclay, LIMSI, codirecteur), M. Nicolas Sabouret (Pr, Université Paris-Sud, Université Paris-Saclay, LIMSI, président), M. Jörg Tiedemann (Pr, Université d'Helsinki, Finlande, rapporteur), M. Loïc Barrault (MC, Université du Mans, rapporteur), Mme Lucia Specia (Pr, Université de Sheffield et Imperial College London, Royaume-Uni, examinatrice), M. Andrei Popescu-Belis (Pr, Haute École d'Ingénierie et de Gestion du Canton de Vaud, Suisse, rapporteur).

Résumé : *Les systèmes de traduction automatique (TA) ont fait des progrès considérables ces dernières années. La majorité d'entre eux reposent pourtant sur l'hypothèse que les phrases peuvent être traduites indépendamment les unes des autres. Ces modèles de traduction ne s'appuient que sur les informations contenues dans la phrase à traduire. Ils n'ont accès ni aux informations présentes dans les phrases environnantes ni aux informations que pourrait fournir le contexte dans lequel ces phrases ont été produites.*

TAL. Volume 59 — n° 1/2018, pages 99 à 105

La TA contextuelle a pour objectif de dépasser cette limitation en explorant différentes méthodes d'intégration du contexte extraphrastique dans le processus de traduction. Les phrases environnantes (contexte linguistique) et le contexte de production des énoncés (contexte extralinguistique) peuvent fournir des informations cruciales pour la traduction, notamment pour la prise en compte des phénomènes discursifs et des mécanismes référentiels.

La prise en compte du contexte est toutefois un défi pour la traduction automatique. Évaluer la capacité de telles stratégies à prendre réellement en compte le contexte et à améliorer ainsi la qualité de la traduction est également un problème délicat, les métriques d'évaluation usuelles étant pour cela inadaptées, voire trompeuses.

Dans cette thèse, nous proposons plusieurs stratégies pour intégrer le contexte, tant linguistique qu'extralinguistique, dans le processus de traduction. Nos expériences s'appuient sur des méthodes d'évaluation et des jeux de données que nous avons développés spécifiquement à cette fin. Nous explorons différents types de stratégies : les stratégies par prétraitement, où l'on utilise le contexte pour désambiguïser les données fournies en entrée aux modèles ; les stratégies par post-traitement, où l'on utilise le contexte pour modifier la sortie d'un modèle non contextuel, et les stratégies où l'on exploite le contexte pendant la traduction proprement dite. Nous nous penchons sur de multiples phénomènes contextuels, et notamment sur la traduction des pronoms anaphoriques, la désambiguïsation lexicale, la cohésion lexicale et l'adaptation à des informations extralinguistiques telles que l'âge ou le genre du locuteur. Nos expériences, qui relèvent pour certaines de la TA statistique et pour d'autres de la TA neuronale, concernent principalement la traduction de l'anglais vers le français, avec un intérêt particulier pour la traduction de dialogues spontanés.

URL où le mémoire peut être téléchargé :
https://tel.archives-ouvertes.fr/tel-02004683

Matthieu LABEAU : mlabeau@exseed.ed.ac.uk

Titre : Modèles de langue neuronaux : gestion des grands vocabulaires

Mots-clés : Réseaux de neurones, modèles de langue, grands vocabulaires.

Title: *Neural Language Models: Dealing with Large Vocabularies*

Keywords: *Neural networks, language models, large vocabularies.*

Thèse de doctorat en informatique, LIMSI, CNRS, Université Paris-Sud, Université Paris-Saclay, Orsay, sous la direction de Alexandre Allauzen (Pr, Université Paris-Sud, Université Paris-Saclay, LIMSI). Thèse soutenue le 21/09/2018.

Jury : M. Alexandre Allauzen (Pr, Université Paris-Sud, Université Paris-Saclay, LIMSI, directeur), M. Pierre Zweigenbaum (DR, CNRS, LIMSI, président), M. Massih-Reza Amini (Pr, Université Grenoble-Alpes, rapporteur), M. Phil Blun-

som (associate professor, University of Oxford, Royaume-Uni, rapporteur), M. Armand Joulin (research scientist, Facebook Artificial Intelligence Research, examinateur), M. André Martins (research scientist, Instituto de Telecomunicaçoes, examinateur).

Résumé : *Le travail présenté dans cette thèse explore les méthodes pratiques utilisées pour faciliter l'entraînement et améliorer les performances des modèles de langues munis de très grands vocabulaires. La principale limite à l'utilisation des modèles de langue neuronaux est leur coût computationnel : il dépend de la taille du vocabulaire avec laquelle il grandit linéairement. La façon la plus aisée de réduire le temps de calcul de ces modèles reste de limiter la taille du vocabulaire, ce qui est loin d'être satisfaisant pour de nombreuses tâches. La plupart des méthodes existantes pour l'entraînement de ces modèles à grand vocabulaire évitent le calcul de la fonction de partition, qui est utilisée pour forcer la distribution de sortie du modèle à être normalisée en une distribution de probabilités. Ici, nous nous concentrons sur les méthodes à base d'échantillonnage, dont l'échantillonnage par importance et l'estimation contrastive bruitée. Ces méthodes permettent de calculer rapidement une approximation de cette fonction de partition. Cependant, elles varient en efficacité, surtout dans le cas des très grands vocabulaires.*

L'examen des mécanismes de l'estimation contrastive bruitée nous permet de proposer des solutions qui vont considérablement faciliter l'entraînement, ce que nous montrons expérimentalement. Nous discutons notamment de l'impact de la distribution de bruit choisie pour échantillonner, ainsi que des autres hyperparamètres impliqués dans l'apprentissage. Ensuite, nous utilisons la généralisation d'un ensemble d'objectifs basés sur l'échantillonnage en tant que divergences de Bregman pour expérimenter avec de nouvelles fonctions objectifs. Enfin, nous exploitons les informations données par les unités sous-mots (caractères ou décompositions morphologiques) pour enrichir les représentations de mots.

À l'aide des méthodes d'apprentissage présentées dans la partie précédente, nous cherchons à entraîner des modèles munis de ces représentations, pour les mots en entrée et surtout en sortie du modèle. Nous expérimentons avec différentes architectures sur le tchèque et nous montrons que les représentations basées sur les caractères permettent l'amélioration des résultats pour les grands vocabulaires, d'autant plus lorsque l'on réduit conjointement l'utilisation des représentations des mots les plus rares, qu'il est difficile d'apprendre.

URL où le mémoire peut être téléchargé :
http://www.theses.fr/2018SACLS313

Caroline LANGLET : langlet.caro@gmail.com

Titre : Analyse des sentiments dans les conversations humain-agent. Vers un modèle des goûts de l'utilisateur

Mots-clés : Interaction humain-agent, agents conversationnels animés, analyse de sentiments.

Title: *Sentiment Analysis in Human-Agent Conversations. Modelling User's Likes and Dislikes*

Keywords: *Human-agent interaction, embodied conversational agent, sentiment analysis.*

Thèse de doctorat en informatique, LTCI (Laboratoire Traitement et Communication de l'Information), IDS (Image, Données, Signal), Télécom ParisTech, Paris, sous la direction de Catherine Pelachaud (DR, CNRS, ISIR) et Chloé Clavel (MC, Télécom Paristech). Thèse soutenue le 26/09/2018.

Jury : Mme Catherine Pelachaud (DR, CNRS, ISIR, codirectrice), Mme Chloé Clavel (MC, Télécom Paristech, codirectrice), M. Bjorn Schuller (Pr, Imperial College of London, Royaume-Uni, rapporteur), M. Thierry Poibeau (DR, CNRS, Lattice, rapporteur), Mme Pascale Sébillot (Pr, INSA de Rennes, IRISA, présidente), M. Dirk Heylen (Pr, Université de Twente, Pays-Bas, examinateur), Mme Marie-Jeanne Lésot (MC, Sorbonne Université, examinatrice), M. Nicolas Maudet (Pr, Sorbonne Université, examinateur).

Résumé : *Cette thèse se situe à la croisée de deux domaines de recherche : celui du sentiment analysis et celui des agents conversationnels animés. Les agents conversationnels animés peuvent être définis comme des personnages virtuels ayant la capacité de converser avec un utilisateur humain. Afin d'accroître les compétences communicationnelles de l'agent, il est important que celui-ci soit doté d'une forme d'intelligence socio-émotionnelle. L'agent doit être ainsi en capacité de gérer des signaux socio-émotionnels, tant du côté de la génération que de celui de la détection.*

Du côté de la génération, de nombreux travaux ont produit des modèles optimisant la production de gestes ou d'expressions faciales pour exprimer soit des émotions soit des attitudes sociales. Du côté de la détection, une majorité des travaux se concentrent sur l'analyse d'indices socio-affectifs non verbaux (expressions faciales, indices acoustiques). Le contenu verbal et les expressions de sentiment qu'il véhicule restent quant à eux encore partiellement exploités. En effet, les rares études intégrant un module de détection des sentiments de l'utilisateur dans le cadre de conversations humain-agent ne prennent pas en compte les spécificités de ce contexte d'interaction.

Pour combler cette lacune, notre travail s'intéresse à l'analyse du contenu verbal produit par l'utilisateur et à la manière dont celui-ci réfère à ou exprime des sentiments, des affects ou des attitudes. Nous en proposons un modèle de détection au cours d'une interaction multimodale et en face à face avec un agent conversationnel animé. Pour

construire ce modèle, deux questions se sont posées à nous. Dans un premier temps, il nous a fallu identifier, au sein de la vaste classe des expressions de sentiment, celles qui apparaissaient comme les plus pertinentes pour l'élaboration des stratégies de communication de l'agent. Dans un second temps, nous avons dû choisir une méthode devant être non seulement opérante pour une analyse à grain fin de ces expressions, mais également adaptable au contexte conversationnel.

Nos contributions s'articulent autour de trois axes. Tout d'abord, nous fournissons un modèle linguistique des expressions de sentiment dans une conversation humain-agent. Trois unités conversationnelles sont considérées : le tour de parole, la paire adjacente et la séquence thématique. Cette analyse met en évidence un certain nombre de caractéristiques nécessaires au développement d'un ensemble de règles de détection. Ensuite, nous proposons un modèle de détection symbolique intégrant des règles sémantiques et des grammaires formelles. Ce modèle repose sur une analyse ascendante des énoncés — du niveau lexical au niveau phrastique — et se concentre successivement sur trois cadres d'analyse : le tour de parole, la paire adjacente et la séquence thématique. Enfin, nous proposons un protocole d'évaluation pour la validation des règles. Grâce à la création de deux plateformes d'annotation, nous avons pu créer deux jeux d'annotations sur deux corpus différents : un corpus de type small-talk *et un corpus de négociation. Les performances du système ont ainsi pu être évaluées par rapport aux références obtenues.*

URL où le mémoire peut être téléchargé :
https://tel.archives-ouvertes.fr/tel-02002580

Elvys LINHARES PONTES : elvyslpontes@gmail.com

Titre : Résumé translingue de textes par compression

Mots-clés : Résumé translingue de textes, compression multi-phrases, multilinguisme, optimisation.

Title: *Compressive Cross-Language Text Summarization*

Keywords: *Cross-language text summarization, multi-Sentence compression, multilingual, optimization.*

Thèse de doctorat en informatique, Laboratoire Informatique d'Avignon (LIA), Centre d'Enseignement et de Recherche en Informatique (CERI), Avignon Université, Avignon, sous la direction de Torres-Moreno Juan-Manuel (MC HDR, Avignon Université, LIA). Thèse soutenue le 30/11/2018.

Jury : M. Torres-Moreno Juan-Manuel (MC HDR, Avignon Université, LIA, codirecteur), M. Stéphane Huet (MC, Avignon Université, LIA, codirecteur), Mme Andréa Carneiro Linhares (MC, Universidade Federal do Ceará, Brésil, codirectrice), Mme Marie-Francine Moens (Pr, KU Leuven, LIIR, Louvain, Belgique, rapporteur),

M. Antoine Doucet (Pr, Université de La Rochelle, L3i, rapporteur), M. Frédéric Béchet (Pr, Aix Marseille Université, LIS, président), M. Guy Lapalme (Pr, Université de Montréal, DIRO, Canada, examinateur), Mme Fatiha Sadat (Pr, Université du Québec à Montréal, GDAC-LIA, Canada, examinatrice), M. Petko Valtchev (Pr, Université du Québec à Montréal, GDAC-LIA, Canada, examinateur), M. Florian Boudin (MC, Université de Nantes, LS2N, examinateur).

Résumé : *The popularization of social networks and digital documents has caused a rapid increase of the information available on the Internet. However, this huge amount of data cannot be handled manually. Natural Language Processing (NLP) deals with interactions between computers and human languages in order to process and analyze natural language data. NLP techniques incorporate a variety of methods, including linguistics, statistics or machine learning, to extract entities, relationships or understand a document. In this thesis, among several existing NLP applications, we are interested in cross-language text summarization which produces a summary in a language different from the language of the source documents. We also look at other NLP tasks (word encoding representation, semantic similarity, sentence and multi-sentence compression) to generate more stable and informative cross-lingual summaries.*

Most NLP applications, including text summarization, rely on a similarity measure to analyze and to compare the meaning of words, chunks, sentences and texts. A way to analyze similarity is to generate a representation for sentences that takes into account their sense. The meaning of sentences is defined by several elements, such as the context of words and expressions, word order and previous information. Simple metrics, such as cosine metric and Euclidean distance, provide a measure of similarity between two sentences. However, they put aside the order of words or multi-words. To overcome these limitations, we propose a neural network model that combines recurrent and convolutional neural networks to estimate the semantic similarity of a pair of sentences (or texts) from both the local and general contexts of words. On a supervised task, our model predicts more accurate similarity scores than baselines by taking greater account of the local and the general meanings of not only words, but also multi-word expressions.

In order to remove redundancies and non-relevant information of similar sentences, we propose a multi-sentence compression method that abbreviates and fuses them in a correct and short sentence that contains the main information. First, we model clusters of similar sentences as word graphs. Then, we apply an integer linear programming model that guides the compression of these clusters based on a list of keywords. We look for a path in the word graph that has a good cohesion and contains the maximum of keywords. Through a series of experiments, we show that our approach outperforms baselines by generating more informative and correct compressions for French, Portuguese and Spanish languages.

Finally, we combine these previous methods to build a cross-language text summarization system. Our system is an {English, French, Portuguese, Spanish}-to-{English, French} cross-language text summarization framework that examines the information in source and target languages to identify the most relevant sentences. Inspired

by the compressive text summarization studies in monolingual analysis, we adapt our multi-sentence compression method for this problem to just keep the main information. Our system proves to be a good alternative to compress redundant parts and to preserve relevant information, without losing grammatical quality. Experimental analysis of {English, French, Portuguese, Spanish}-to-{English, French} cross-lingual summaries indicate that our approach significantly outperforms the state of the art for all these languages. Besides, we apply cross-language summarization and discuss its role in two applications: microblog contextualization and spech-to-text summarization. In the last case, our method still achieves better and more stable scores, even for transcript documents that have grammatical errors and missing information.

URL où le mémoire peut être téléchargé :
https://hal.archives-ouvertes.fr/tel-02003886